# Optimisation énergétique de processus de traitement du signal et ses applications au décodage vidéo

Erwan Nogues

Thèse

UNIVERSITE BRETAGNE LOIRE

**THESE INSA Rennes**
*sous le sceau de l'Université Bretagne Loire*
pour obtenir le titre de
**DOCTEUR DE L'INSA RENNES**
*Spécialité : Traitement du Signal et de l'Image*


présentée par
Erwan Nogues
**ECOLE DOCTORALE : MATISSE**
**LABORATOIRE : IETR**


# Energy Optimization of Signal Processing on MPSoCs and its Application to Video Decoding

**Thèse soutenue le 02.06.2016**
devant le jury composé de :

**Emmanuel Casseau**
Professeur à l'Université de Rennes 1 / *Président*
**Bertrand Granado**
Professeur à l'Université Pierre et Marie Curie / *Rapporteur*
**Cécile Belleudy**
Maître de Conférences HDR à l'Université de Nice Sophia Antipolis / *Rapporteur*
**Dominique Ginhac**
Professeur à l'Université de Bourgogne Franche-Comté / *Examinateur*
**Eduardo Juárez**
Associate professor à l'Université Polytechnique de Madrid / *Examinateur*
**Eric Senn**
Maître de Conférences HDR à l'Université de Bretagne Sud / *Examinateur*
**Maxime Pelcat**
Maître de Conférence à l'INSA Rennes / Co-Encadrant
**Daniel Ménard**
Professeur à l'INSA Rennes / Directeur de thèse

# Energy Optimization of Signal Processing on MPSoCs and its Application to Video Decoding

Erwan Nogues

# Contents

# Acknowledgements

First, I would like to thank my thesis director Pr. Daniel Ménard and my advisor Dr. Maxime Pelcat for their guidance, their support, and their trust for the last years. Thank you for giving me the opportunity to pursue a PhD on such an interesting topic and in such a motivating and friendly working environment. Daniel, thank you for your advice during all this work, your great expertise and your strategic orientations to move forward. Maxime, thank you for your open-mindedness at any time during this PhD, your capability to be source of proposals and being always ready to help. All and all, we made a great team.

I want to thank also Pr. Jean-François Nezan to encourage me to start this PhD and put everything together so I could meet Daniel and Maxime. It's been a long trip since Sofia, Studentskigrad, July 1998.

I want to express my gratitude to Pr. Bertrand Granado and Dr. Cécile Belleudy for being part of the PhD committee and for taking time to review this thesis. I also want to thank Pr. Emmanuel Casseau for presiding the PhD committee and Pr. Dominique Ginhac, Dr. Eduardo Juárez and Dr. Eric Senn for being members of the PhD committee.

I also would like to thank Pr. Olivier Desforges and Pr .Luce Morin for welcoming me within the IETR Image group. More generally, I want to thank all the members of the image group for making me feel part of the team from the beginning.

Special thanks to all my very talented office-mates during this work, I learned a lot: Karol Desnos, Julien Heulot, Antoine Lorence, Judicaël Menant and Alexandre Mercat. Thanks also to Wassim Hamidouche, Erwan Raffin and Mickael Raulet for their support on HEVC. Thanks to Alexandre Sanchez for our interesting talks about Malotru. Also, thanks to Frédéric Garesché for his IT support and many thanks to Corinne Calo, Aurore Gouin and Jocelyne Trémier for their administrative support.

I am also very grateful to all the students that helped me with their hard work: Romain Berrada, Louis-Paul Cordier, Glenn Herrou, Morgan Lacour, Kévin Reuzé, Ladislas Robin and Simon Urosevic.

I would like to thank all the people from the Green Video project, especially Xavier Ducloux. I really appreciated our technical discussions.

My PhD also started very well with a good kick-off with Simon Holmbacka, my Linux friend for Finland.

I would like to thank Pr. Arnaud Martin and Dr. Mouloud Kharoune for giving me the opportunity of my first teaching experience at IUT Lannion. Teaching was a primary goal of doing a PhD.

Finally, thanks also to my former and new colleagues for their support: Anis, Christophe, Daniel, Fred, Jean-François, Jean-Yves, Landry, Laurent, Pascal (x2), Pierre (x2), Pierrick, Michel, Nicolas, Sébastien, Thierry, Yannick.

Je tiens aussi à remercier ma famille : Yvon, Marie-Annick, Sébastien, Carole, Maurice et Marie-Madeleine pour leurs encouragements et leur soutien durant ces années de thèse.

Enfin, je ne remercierai jamais assez ma chère compagne, Aude, qui m'aura encouragé à démarrer ce projet *pas comme les autres* dès que l'idée a germé dans mon esprit. Et merci de m'avoir toujours soutenu. Merci aussi à mes deux petits fans, Louise et Martin. On aura passé du bon temps à discuter tous les matins sur le chemin de l'école. Au final, cette expérience restera unique.

CHAPTER 1

# Introduction

## 1.1 General Context: Video in Mobile Networks

More than ever, we live in a society where images are everywhere. In particular, our daily lives have been invaded by images and videos. The time when the only way to watch a content was to sit in front of a television is over. Modes of video utilization have been much disrupted over the last fifteen years. This progress was made possible thanks to the availability of efficient coding techniques, communication systems capable of supporting large bandwidth, and highly performing electronic systems. All these elements combined supply a virtuous circle where improved coding techniques enable better contents to be transmitted, increased communication bandwidth makes it possible to address more people or more advanced features, and new generations of electronic components provide better conditions to watch video. All in all, people can now access video content just about anywhere and on a wide range of devices.

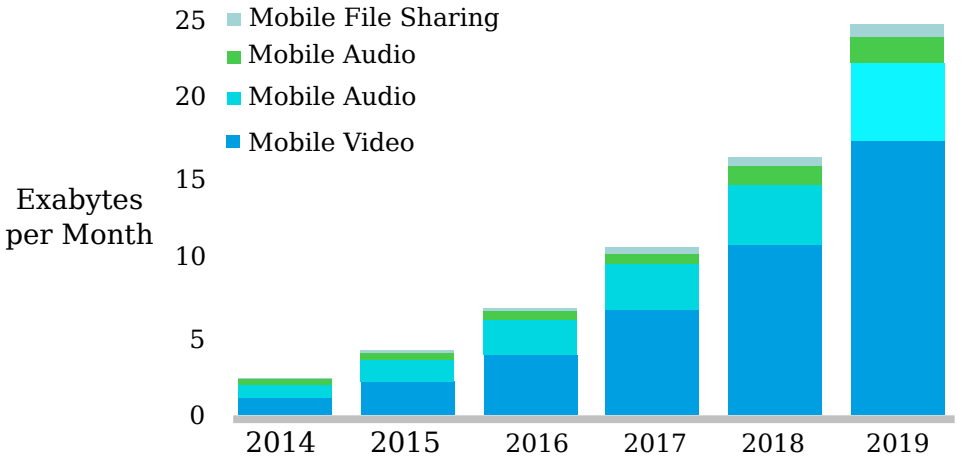


**Figure 1.1:** *Cisco analysis of data exchange of mobile networks [Cis15]*

As a consequence, networks are now primarily used to supply video content to users. Cisco regularly publishes the data usage within the worldwide network [Cis15]. As depicted in Figure 1.1, data exchange is exploding together with the increasing number of supported video applications.

### 1.1.1 The Video Eco-system

For a long time, the video eco-system has been limited to broadcasting video contents. These contents were traditionally centralized on servers owned by broadcasters. This situation drastically changed with the arrival of new uses cases. The production of video content is much easier today than it used to be. Video transmission is made possible by advanced network capabilities and video processing can be performed on a wide range of devices. As depicted in Figure 1.2, the content is not only focused on TV networks and studio production. Over-The-Top (OTT) service providers are likely to have a major role in the new eco-system thanks to IP-based media services. Furthermore, video decoders are embedded onto a wide range of devices: from TV to mobiles or tablets. The requirements are then different if the video content targets high definition televisions or midrange smartphones.

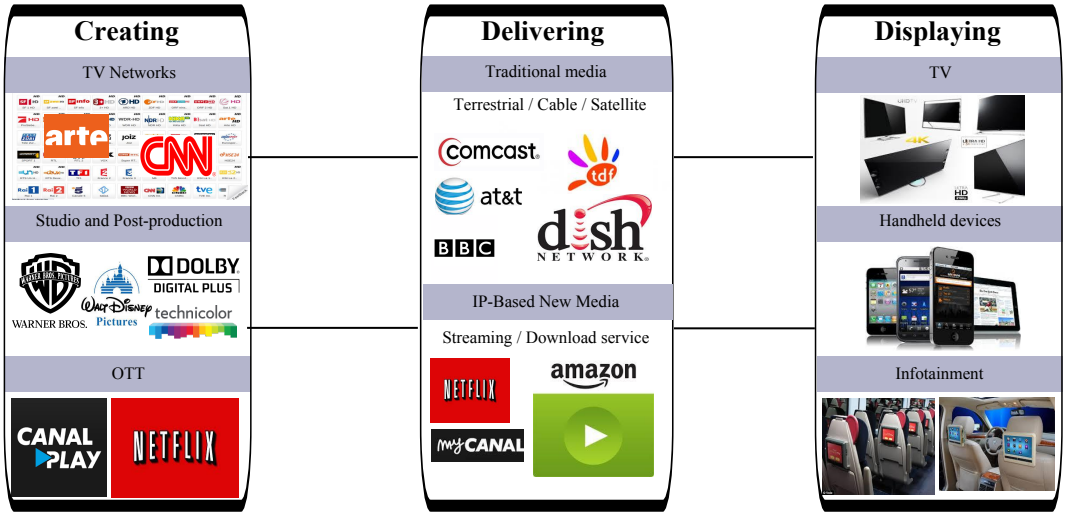


**Figure 1.2:** *The Video Ecosystem*

The final cut is done by the user experience. The user requirement is to access video content with a minimum of Quality of Service (QoS) or Quality of Experience (QoE). The metrics for the evaluation of quality are numerous: image resolution, sampling frequency of the video, latency or up-time of the device...

### 1.1.2 An Increasing Use of Mobile Devices for Video Applications

The video eco-system is now driven by the utilization of video in mobility. This fact is mainly due to the increasing use of video sharing applications (e.g. Youtube, Netflix), social networks (e.g. Facebook, Instagram), IPTV on mobile and video-conferencing. These new practices, combined with the recent advances on network capabilities, induce an exponentially growing traffic. However, what characterizes mobile devices is their capacity of being autonomous and guaranteeing an utilization in mobility. The challenges for these devices are to support ever more complex features (e.g. new video codecs), to remain extensible to

new features and to function under the very strict power envelope offered by batteries. For example, Carollet *al.* report that the processing resources can consume more than 60% of the total power consumption to play back an H.264/AVC video [RJS+10]. Software-based solutions are some of the possible alternatives for video processing. Indeed, they offer flexibility to adapt devices to new features and video codecs. However, it is a difficult task to design software-based video decoders that are also energy efficient.

### 1.1.3 MPSoC Architectures: Making High Performance Mobile Systems Possible

Video processing is achieved with sophisticated algorithms that demand a lot of computational power. The common way to improve system performance accordingly is to increase the processing frequency and *process more* in the same period of time. However, increasing processor clock frequency is only possible to a certain extent because of power dissipation. The phenomenon is known as the *power wall* and pressures the semiconductor industry to switch to multicore architectures. Such a solution can provide more computational capabilities for a given power budget. Thanks to this technological advance, applications such as video decoding, that were traditionally hardwired, can now be processed in software [BAMG+13, CAMJ14, CAML+13a, RNH+15, CPG+13].

Embedded systems benefit from the multicore architecture enhancements to improve their computational capabilities. As of today, modern designs of portable devices use multicore architecture to support high-end features [exy13].

### 1.1.4 Enhanced Software-based Energy Management in MPSoCs

Energy efficiency is critical to handheld devices. On top of providing high performing processors, the modern SoCs also provide means to control the power consumption of the device. At a synthetic level, the power consumption of an electronic device is composed of two elements, the static power and the dynamic power.

Firstly, the static power does not depend of the processing load. It depends on the circuit fabrication technology and area (number of transistors). To reduce it, the processor shall enter into sleep modes when idling. Secondly, the dynamic power is determined by the processing activity together with the voltage supply and the processing frequency. Scaling down the frequency of operation and the voltage supply reduces the dynamic power. On the one hand, mechanisms like Dynamic Power Management (DPM) are now available in SoCs to reduce the static power by shutting down blocks or clocks during idle periods. On the other hand, Dynamic Voltage Frequency Scaling (DVFS) adapts dynamically both the frequency and the supply voltage. These mechanisms are made available at the user space level to design low power embedded software implementations.

### 1.1.5 A Wider Range of Supported Applications in Mobile Devices - HEVC: a Promising New Standard

From the application point of view, the challenge lies in providing high compression rates for the transfer of videos. Motion Picture Expert Group (MPEG) introduced in 2013 the High Efficiency Video Coding (HEVC) standard that improves the bitrate by 50% compared to its predecessor H.264 at the same video quality. As an example, this surplus of bandwidth can be used by service providers to offer the video services to more devices. The video streaming case is the typical example of how the video offer can make the most

of the latest technological advances. It is implemented via the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard [Sod11].

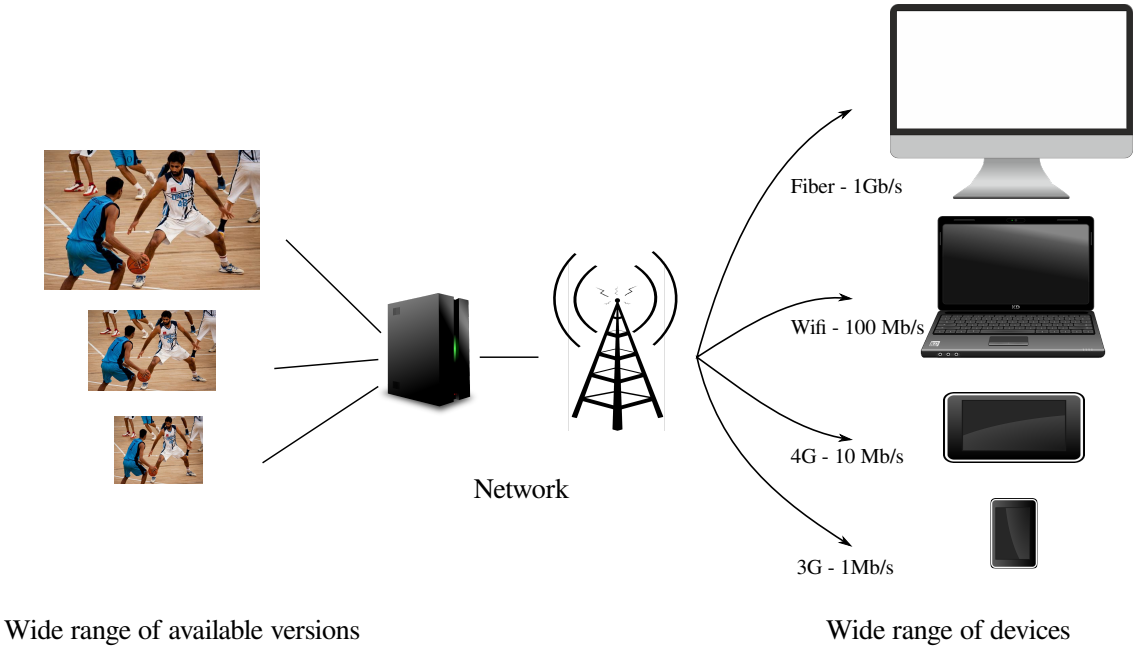


**Figure 1.3:** *Example of streaming applications using the diversity of the network and addressing a wide range of devices*

As depicted in Figure 1.3, several versions of the same content are available at the MPEG DASH server level. Depending on the network capabilities of the physical layer and on the features supported by the device, the initial content can be scaled down to adapt its payload. For example, the resolution can be resized, the display frame rate can be reduced or the quality can be lowered. All these features are intrinsically supported by HEVC. The feasibility of HEVC decoding on a wide range of devices is thus essential to the adoption of the standard. MPEG paid a special attention at the design phase of HEVC to take into account its complexity. Therefore, software based implementations like OpenHEVC [Ope] are available. They enable quick market penetration of this new standard with no hardware upgrade.

To be widely adopted, a new standard such as HEVC needs to prove itself in terms of energy efficiency. HEVC shall use efficiently all the latest technical advances of the underlying platform, i.e. parallelism, multicore architecture and low power design techniques. On top of that, MPEG acknowledged that low power video decoding is a real challenge. Therefore, the standardization body launched an ad-hoc working group also called Green Metadata [Gre13] to offer means to reduce the power consumption of the video decoding.

## 1.2 Objectives of the Thesis

The main objective of this thesis is to analyze how the low power management techniques embedded in modern System-on-Chip (SoC) can be utilized together with the application requirements to meet user expectations. Even though numerous techniques exist on Multiprocessor SoC (MPSoC) to design applications in a low power fashion, using them on a complex use case is far from easy.

By using a top-down approach as shown in Figure 1.4, this thesis aims at providing a comprehensive analysis of the energy efficiency at the different steps of the design phases. At a low level of abstraction, we present how data parallelism can be exploited with Single Instruction Multiple Data (SIMD) operators. Then we present how task level parallelism and multi-threading can be efficiently mapped onto MPSoC to reduce the energy. Finally dynamic frequency scaling can be used at a top level set-up to use efficiently the dynamic power reduction.

Another objective is also to analyze how the low power management techniques can be used efficiently for a given application and how application characteristics shall be taken into account. The analysis shall result in solutions to implement the energy efficiency paradigm that gathers real-time QoS and overall QoE requirements.

The targeted application is the cutting edge video coding standard, HEVC. This class of applications faces a real implementation challenge on embedded systems. Indeed, it requires much computing resources, and consequently energy, to achieve real-time performance.

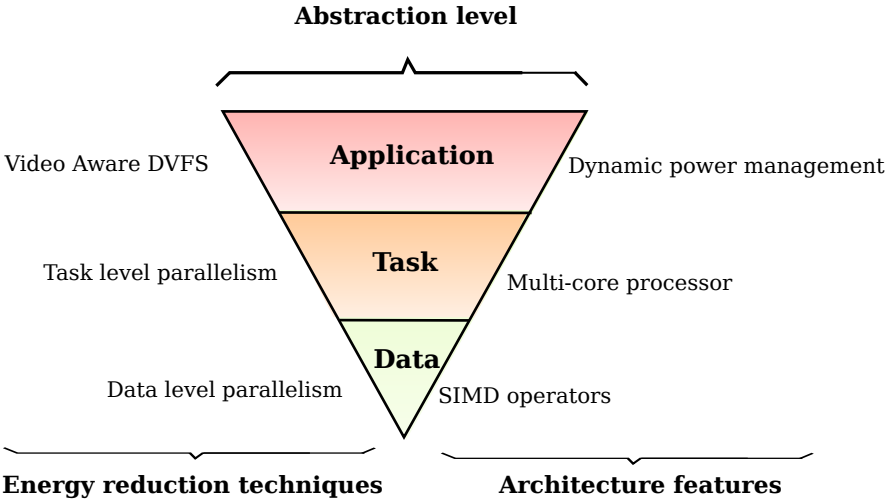


**Figure 1.4:** *Energy efficient design methodology*

## 1.3 Contributions of the Thesis

Based on the scheme depicted in Figure 1.4, the main contributions of this thesis are:

1. A framework offering **Rapid Prototyping of Signal Processing Applications**. We propose here to characterize a MPSoC in terms of power and energy [HNP+14]. We introduce an energy model that takes into account globally the SoC **dynamic frequency scaling capabilities, deep sleep modes and parallelism levels** [NPM+16]. Moreover, by injecting the application characteristics in the platform energy model, we propose to solve the energy minimization problem under real-time constraints. The proposed formulation of the problem exploits the convex properties of the power measurements guaranteeing to find an optimal solution in a polynomial time [NPM+16]. Using a model of a State-of-the-Art architecture, we find the most energy efficient operating point of the MPSoC. The modeling proposition is also validated on a real SoC with the HEVC application.

2. A real-time **low power HEVC decoder using efficiently the DVFS capabilities of a General Purpose Processor (GPP)-based MPSoC**. Typical tech-

niques fail to assess the high variations of the *per-frame* processing. We propose here to use the DVFS capabilities of the MPSoC to reduce the power consumption. We show how our proposed implementations can achieve close-to-optimal performance with tight real-time requirements [NBP+15, RNH+15]. We propose two solutions: either *intrusive* with the inclusion of DVFS inside the HEVC decoder or *non-intrusive* where only the real-time requirements of the system are required. As a conclusion to this contribution, the results prove that HEVC can be implemented in real-time on an on-the-shelf GPP and therefore be deployed on a wide range of consumer electronic devices with no dedicated circuit.

3. A modified **HEVC decoder with tunable image quality**. This last contribution proposes to use the error tolerance property of video applications. Indeed, multimedia applications are inherently error tolerant since the final applications quality is left to the Human Visual System (HVS). Therefore, strict exactness may not be required and an imprecise result can be sufficient. We propose a framework based on approximate computing to save energy while decoding HEVC. We dive into the decoder itself from an algorithm signal processing point of view. The induced quality loss can be controlled at the device level according to the desired energy/quality tradeoff. It also provides a very fine grain tuning capability [NHP+14, NRPM15c, NRPM15a].

All these contributions were developed as part of the FUI project called Greenvideo [FUI] and were also promoted to the MPEG GreenMetadata Ad-Hoc group [FDM+15] through several contributions [NDR+14, NRQ+15, RNQ+15, RLP+15].

## 1.4 Outline

The thesis is organized in two parts: Part I introduces the background, concepts and research issues, and Part II presents and evaluates the contributions of the thesis.

In Part I, Chapter 3 recalls the context and the motivations of the thesis. It particularly focuses on the video use case and presents the different parts of a complete video processing chain. Then, the different techniques of low power design are described as well as their influence on the different sources of power consumption. Chapter 3 presents the related work on low power decoders and especially for HEVC standard. It emphasizes the importance of coupling the platform low power techniques with the application requirements.

In Part II, a new energy model for MPSoC is introduced in Chapter 4. The model is used in the context of a rapid prototyping framework. Performance evaluations for offline HEVC decoding are given. Following the design flow of Figure 1.4, Chapter 5 takes into account the real-time characteristics of the HEVC decoders to propose efficient low power implementations. The performances are thoroughly evaluated on a wide range of video sequences. Chapter 6 introduces an approach based on approximate computing and proposes an evolution of the standard HEVC decoder that offers large power savings with a controlled decoding distortion. This approach is presented within the MPEG GreenMetadata initiative and motivates the use of a metadata to control the tradeoff between the decoder power consumption and the decoding quality.

Finally, Chapter 7 concludes this work and opens potential research directions for future works.

# Part I

# Scope and Motivations

CHAPTER 2

# Embedded Video Context

## 2.1 Motivation for Energy Optimization

When using a portable device for video applications, any user wishes the battery life to last longer. Hence, there is already a large demand for increased battery life on devices and this phenomenon keeps accelerating with the success of media applications. At the heart of embedded processing systems, video applications such as streaming (e.g. Video On Demand (VOD) service), conferencing and sharing use a major part of the energy budget.

In this section, the main features of video applications are described first. Then the scope of analysis is focused on the decoding part, especially on electronic portable devices. Finally the performance of these devices is analyzed through the power and energy perspective.

## 2.2 Video Applications

### 2.2.1 The Video Chain and its Energy Consumption

The video chain is split into several components from the capture to the rendering, as presented in Figure 2.1. In this section, we detail the different elements of this chain.

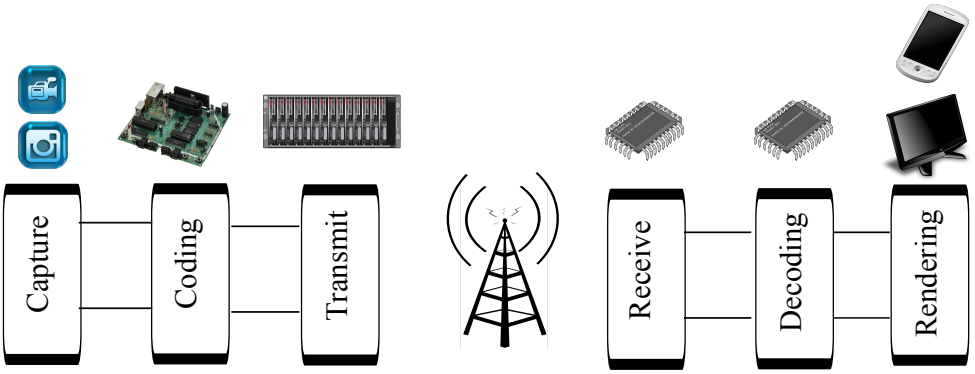


**Figure 2.1:** *Components of the video chain*

### Video Capture

Initially, video cameras were developed for the television industry and were capable of capturing scenes in motion pictures. Over the last decades, these sensors have evolved along with the type of applications. Associated with them, a wide a range of services are now possible. Because the camera is located at the very beginning of the video chain, the type of camera is determined by the targeted type of application:

- professional video cameras are high-end devices that create electronic images in movement. Developed originally for a use in television studios, they are now also used for other applications like music or personal videos.
- camera recorders combine a camera with a video recorder. These cameras are mobile and are used in many fields like live journalism, movies, etc. With the arrival of digital video cameras, camera recorders have an in-built digital recording device.
- Closed-circuit television (CCTV) generally uses zoom cameras for security purposes, and monitoring purposes. CCTV cameras are generally small, discreet and robust to hostile environment.
- webcams provide images in real-time to a computer or a computer network. The video stream may be saved, viewed or sent on to other networks via systems such as the internet, and email as an attachment.
- camera phones are generally designed to be small and simpler than separate digital cameras. Their usual fixed-focus lenses and smaller sensors limit their performance in poor lighting.
- stereo cameras embed two lenses with a dedicated image sensor for each lens. They are generally used to capture 3D videos. The distance between the two cameras are generally similar to the distance between human eyes.
- scientific cameras are devices dedicated to a targeted application. The most common applications include astronomy with on-board devices, robotics, or medical devices. They are designed to work in a large, and non-visible, spectrum. For example, infrared is used for night vision and hyperspectral cameras are used to detect tumors in medical applications.

From an energy point of view, the list above can be sorted by type of power supply. For example, camera phones or on-board cameras are supplied by a battery and strongly necessitate an energy efficient design. It is also a fast growing market as they can be embedded barely anywhere (carried by people or drones).

The case of a mobile phone camera can be considered as one the most sensitive cases. Thanks to the recent advances in electronic devices, Complementary Metal Oxide Semiconductor (CMOS) image sensors require only a few mWatts [Fos94]. This power consumption can be considered as negligible compared to the other entities of the system.

### Video Transmission

#### *Current Status of the Video Transmission Energy Consumption*

Within the production - distribution - consumption chain of multimedia content, the distribution segment represents something significant in terms of energy consumption. Digital terrestrial TV (DTTV) is specifically considered in this section.

The reduction of the consumption of this segment is becoming a real challenge for broadcast operators since they switched from the analog to the digital domain in the last decade. For example, 15000 transmitters are deployed in France with transmitted power ranging from a few Watts to several kWatts. DTTV uses Coded Orthogonal Frequency Division Multiplex (COFDM) multicarrier waveforms as modulation, characterized with a high Peak to Average Power Ratio (PAPR). Therefore, the power amplifiers cannot be used at their saturation level and the average power of the transmitted signal is about 12 dB below the peak power. The resulting energy efficiency ratio for a transmitter is between 10% and 20%, for first generation digital transmitters delivering from 500 Watts to 10 kWatts [NT].

***Reducing the power consumption of the broadcast network***

From about 2010, the transmitter manufacturers have been working a lot on reducing the energy consumption of their products, under the pressure of the network operators. Several directions are followed in that purpose. After the successful deployment of digital TV, network operators shifted their priority from coverage to energy efficient of the transmitter. Several solutions were developed for that purpose.

- **the reduction of the energy consumption of the digital part of the transmitter**. A transmitter is composed of a digital modulator plus some monitoring facilities that control the whole transmitter and its cooling system. The energy consumption of the digital modulator is independent from the antenna output power of the transmitter. The digital modulator is implemented with hardwired logic and programmable technologies like FPGAs and CPUs. These technologies continuously evolve towards more energy efficient solutions. The digital modulator part of the transmitter energy is then reduced but it represents a limited share of the global energy consumption. The benefits on the whole transmitter energy consumption is thus limited, except for low power transmitters (below 1 kWatts).

- **Reduction of the energy consumption of the power amplifiers.** This is the main part of the consumption of a high power transmitter. It requires 1 kWatts or more of power. As presented before, the low efficiency of this part results from the high PAPR of the modulated signal and the inherent low efficiency of the analog RF power amplifiers. Manufacturers seek low power designs with the following items. At first, different approaches exist for reducing the PAPR [VEWT96, JW08] of the multicarrier waveforms, including standardized ones like in the latest second generation DTTV standards, such as DVB-T2. Tradeoffs have to be made as reducing the PAPR introduces noise and degrades the signal quality. This PAPR reduction is performed in the digital modulator, as it is closely coupled to the waveform generation function. It brings a gain of a few percents in power amplifier efficiency.

  Another option is to linearize power amplifiers. Adequate digital pre-processing compensates the inherent non-linearities of power amplifiers [SS83, SM98]. Using linearized amplifiers, the average operating power can be raised, thus increasing the energy efficiency, while maintaining the same signal quality at the output. This pre-processing is digitally performed at the output of the modulator. It is using a return channel so that the processing can dynamically adapt itself to the characteristics of the power amplifier, including variations linked to the power supply, the temperature, and even the memory effect of the power transistors. This approach brings a gain of a few more percents in power efficiency.

Finally, new amplifier technologies have better energy efficiency. Traditional Class AB amplifiers have a limited efficiency ratio (typically 25%). Alternative amplifier architectures, such as Doherty ones, bring higher efficiency (up to 40%) and are widely adopted in the latest generations of transmitters [IWC+01]. One drawback is the limited bandwidth of such architectures. Most implementations are not wide band (460 - 860 MHz). Other amplifier technologies are used in lower power systems such as the cellular phone base stations: Envelop Tracking power supplies [WYK+05] and Switched Mode amplifiers [MBM+98]. They are not really usable in the broadcast environment, where many low power amplifier modules must be parallelized to get the required power level. Implementation complexity would then be far too high.

By following all these technical directions at the same time, transmitter manufacturers are improving significantly the efficiency of their products. The latest high power transmitter now reach 40 to 45 % efficiency, which is typically twice the efficiency met 10 years ago.

### Video Rendering

Video rendering is a mirror image of video capture. It also addresses a wide range of components that are tightly coupled with the user-defined applications. From cinema large screens to smart watches, power consumption constraints are drastically different. Within the scope of handheld devices, the rendering and display elements utilize a large share of the overall power budget. Ma *et al.* [Sam12a] showed indeed that in a phone reading streaming video, 400 mWatts are needed for the display, 300 mWatts for the video decoding, 250 mWatts for the idle part, and 300 mWatts for downloading the video. The measurements were done on a mid-range smartphone when a video sequence is being played.

Depending on the underlying technologies, several proposals were formulated to reduce the power consumption of the display device. Chang *et al.* [CCS04] proposed back light scaling for LCD systems. The induced distortion is compensated by an appropriate image mechanism to keep the perceived image contrast as close as possible as the original one. Shin *et al.* [SKCP11] propose a new principle applicable to the OLED technology adopted in newer equipments. Power consumption is then improved but it highly relies on the hardware technology used by the end device.

### Discussion

In the above approaches, the reduction of energy consumption in the video chain is obtained by two different techniques. A first technique consists in improving the overall efficiency of the process. It is typically the case in the transmission part. The purpose of transmission energy optimization techniques can be rephrased as *to do the same for less energy*. The obtained improvements rely on technological advances and are generally independent from the video use case.

The second technique uses a singular property of the video context. Indeed, from a unique original scene or content to be captured, the final content that is delivered can take various forms. This so-called *scalability* is possible in different components of the chain from the capture to the delivery. Obtaining the correct energy balance in the chain from end to end is then a key challenge and it shall be emphasized that with the explosion of video capable devices, one source of video is likely to address a wide range of terminals with different features and capabilities.

In this thesis, we focus on the video decoding component of the chain and provide guidelines to reduce decoder energy consumption. Reducing of a few Watts the power consumption of billions of devices can have have a major impact at a large scale. The objective of our study is to propose new techniques for two objectives: improve the video processing efficiency and adapt the content to the device. The cornerstone of the decoding system is the compression technology. The rest of this section describes the main features of it.

### 2.2.2 The MPEG Video Compression Standards

The most critical part of a video chain is the compression part since it provides means to transmit complex video contents to various supports.

#### Techniques of Modern Video Coding

Video coding belongs to the source coding theory and aims at reducing the data size of the video. The final goal of video coding is to make content broadcasting easier [SS95]. For example, a typical bandwidth needed for transmitting a high definition video content is 600 Mbits/sec. After compression, the bandwidth can be reduced to 7 Mbits/sec.

The principle of video coding is based on the high statistical redundancy of video sequences both in time and in space domains. The basic statistical property that compression techniques leverage on is the correlation between pixels. This correlation appears in both space (adjacent pixels in an image are similar), and time (colocated pixels in past and future frames are similar to the current pixel).

Thus, it is assumed that the color of a particular pixel of an image can be predicted from the neighboring pixels of the same image (using intra-frame coding techniques) or pixels of a neighboring image (using inter-frame techniques). Intuitively, it is obvious that in certain circumstances, for example, when changing plan within a sequence, the temporal correlation between neighboring pixels between images is small or zero. In this case, techniques that use the spatial correlation are the most appropriate. However, if the correlation between pixels in adjacent frames is large, for example, in the case when two consecutive frames have a similar or identical contents, it is preferable to use a so-called inter coding technique which uses the temporal prediction. In a typical compression scheme, an adaptive combination between the two predictions (temporal and spatial) is used to achieve significant data compression.

Because implementing the complete video chain is a complex task, and to ensure a tight relation between the encoding and the decoding processes, a working group gathering video experts was created to standardize efficient compression systems within a single set of specifications. This group is called Motion Picture Expert Group (MPEG) and brings together many experts from the electronic component industry, information technology and telecommunications.

#### General structure of MPEG

MPEG meetings began in the late 80's to develop a first standard. MPEG then quickly produced new series of international standards, including the still popular MPEG-2, primarily targeting applications related to digital TV. The basic framework of the transform-based motion-compensated decoders, followed by all MPEG decoders, is depicted in Figure 2.2.

In the first step, the entropy decoder extracts the different syntax elements from the video stream using arithmetic coding, after which the residual data (non-predicted data) are

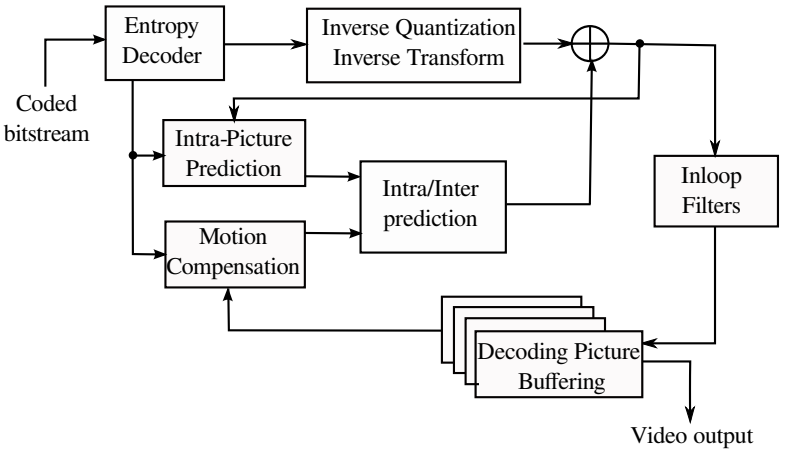


**Figure 2.2:** *Basic modules of the video decoder*

dequantized and transformed using an inverse Discrete Cosine Transform (DCT) process. The prediction of the frames is then applied, and can be either of intra- or inter-frame type depending on the input bitstream parameters. In the case of the inter-frame prediction, a prediction is computed based on the previously decoded pictures and estimates of the motion vectors between pictures at a fractional pixel level. Finally, the in-loop filters are applied on the reconstructed data to reduce potential artifacts due to compression and increase picture quality.

H.264 is introduced here as an canonical example of an MPEG codec. H.264/AVC is based on this framework and is widely adopted thanks to its bitrate / distorsion performance [OBL+04]. Amongst its main features, H.264/AVC uses a Context-adaptive binary arithmetic coding (CABAC) or Context-adaptive Huffman variable-length coding (CAVLC) as entropy coding scheme. CABAC is an arithmetic coding producing excellent results in terms of compression, but at the cost of a complexity higher than CAVLC. CAVLC is an adaptive Huffman coding of variable length, which is a less complex alternative to CABAC coding.

The motion estimation and compensation can be performed over several already coded reference images. The choice of the reference image occurs at the macroblock and sub-macroblock levels. This allows the decoder to use in some cases up to 16 reference images and up to 4 different references for the same macroblock. For certain types of scenes, such as rapid changes, repetitive flashes or recurring scenes, this scheme provides a significant reduction of the actual bitrate. A motion compensation can use seven different block sizes ($16 \times 16$, $16 \times 8$, $8 \times 16$, $8 \times 8$, $8 \times 4$, $4 \times 8$, $4 \times 4$) to define moving zones with accuracy. The precision for motion compensation is based on quarter-pixel for luma data. Motion compensation of chroma values uses the same principle with an eighth-pixel precision.

Finally, an anti-block filter or deblocking filter is carried out in the coding loop and operated on the $4 \times 4$ blocks so as to reduce artifacts due to block processing.

The output frames are traditionally coded with 8-bit data format on 3 components called YUV (one luma, Y and two chrominance, UV components). To increase the perceived quality, higher bit-depths (e.g. precision of 10 bits) have been proposed more recently. These methods are called High Dynamic Range (HDR) [RHD+10].

### Legacy HEVC

Video coding standards such as High Efficiency Video Coding (HEVC) manipulate images represented as fields of color pixels. Next section introduces the different types of possible image organizations.

#### *Image Organization*

An image or a video frame is a rectangular region containing pixels (pels, samples) to be compressed. One of the characteristics of natural images is the strong similarity between neighboring pixels. Indeed, unless they belong to different objects, pixels are likely to be strongly correlated.

There are three components per image: luma (Y), chroma blue (Cb, U), and chroma red (Cr, V). The Y component is associated to gray levels, which enables the Human Visual System (HVS) to distinguish shapes. The later two components (Cb and Cr) represent a transition from gray towards respectively blue and red.

The selection of (Y,Cb,Cr) triplets over the primary (Red,Green,Blue) triplets captured by the HVS has to do with the fact that the HVS is more sensitive to gray levels, and less sensitive to colors. As a consequence, chrominance values can be subsampled without loosing perceptual information.

Image compression standards, including MPEG HEVC, use 4:4:4, 4:2:2, 4:2:0, or 4:0:0 subsampling patterns. The former specifies full resolution where all chrominance samples are kept. The later indicates a monochrome image (no chrominance samples are kept). 4:2:2 and 4:2:0 indicate that a subset of chrominance values is kept. The 4:2:2 pattern is for full horizontal resolution and half vertical resolution. 4:2:0 specifies half resolution in both directions. The subsampling patterns are illustrated below Figure 2.3.

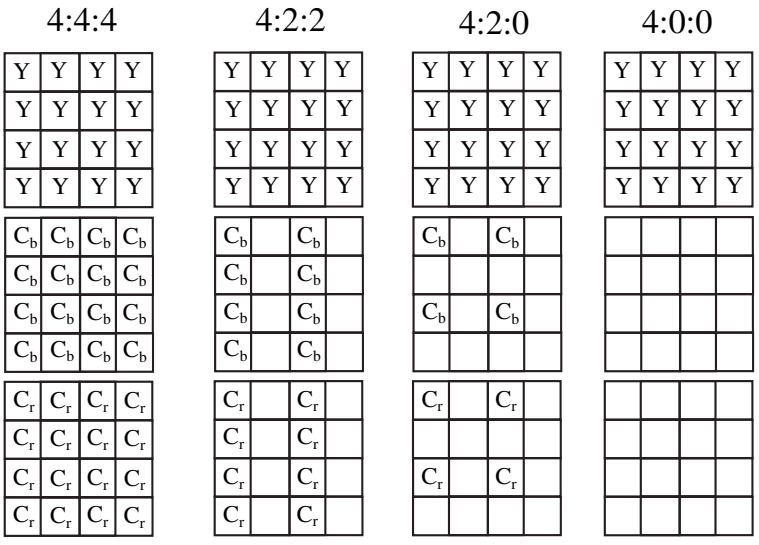


**Figure 2.3:** *Image subsampling patterns*

When 4:4:4 sampling is used, each component may be coded independently as if three different images were coded independently. This means that the samples of a component will be processed before the next component is coded. The alternative is to interleave the components; block-by-block, the Y, Cb, then Cr values are coded. Using the MPEG HEVC codec, the components are processed using either 4:2:2 or 4:2:0 subsampling.

Finally, the bit-depth is defined for all the different components. The current range varies from 8 to 14 bits. The bit depth for luma pixels can differ from the bit depth of chroma pixels.

### *Main improvements*

HEVC is the latest video compression standard from MPEG [SOHW12a]. The first version of HEVC was finalized in January 2013 by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) under a partnership known as the joint collaborative team on video coding (JCT-VC). The HEVC standard brings a gain of up to 50% in terms of subjective video quality [OSS+12b] with respect to the H.264/AVC high profile [WSBL03].

Moreover, the second version of the HEVC standard embeds new coding functionalities, including an efficient coding of video at high bit depth (up to 14 bits) and with high color format fidelity: 4:2:2 and 4:4:4. The HEVC rate-distortion gain together with these new functionalities foster the proliferation of new video services such as video transmission at ultra high quality over broadband and wireless networks for both fixed and mobile devices.

Compared to H.264/AVC, HEVC advances lie in:

- in-built parallelism: compared to H.264/AVC where parallelism was thought afterwards, HEVC standard includes different levels of parallelism: independent slices, tiles and a *wavefront* mechanism. These levels are defined in the HEVC standard to simultaneously process multiple regions of a single image [SSM14].
- intra-prediction is improved for video and still images. There are two categories of intra-predictions: angular prediction and planar prediction [LBH+12].
- inter-prediction is improved and generalized compared to H.264/AVC. It uses advanced motion vector prediction based on motion vector competition [HOB+12].
- in-loop filtering: HEVC contains a deblocking filter as in H.264/AVC. It also adds a new filtering stage called sampling adaptive offset (SAO) which provides both subjective and objective improvements [NBF+12].
- improvements of the CABAC performance to approach ever closer to the inherent entropy coding bound [SB12].

### *Profiles*

HEVC addresses a wide range of applications. To support the application variations without altering the core specifications, HEVC supports several profiles similarly to prior standards. Three profiles are defined in the first version of the standard.

- **Main profile** is used in the typical applications using 8-bit data per sample with a luma component for the brightness and two chroma components that have half the luma resolution both horizontally and vertically (4:2:0),
- **Main Still Picture profile** is used for still photography or snapshot extraction from video sequences. It is a subset of the capabilities of the Main profile.
- **Main 10 profile** is used for to support 10 bits per sample of the decoded frame. This profile aims at integrating the main HDR improvements.

As stated before, the main profile is widely used in various systems like TV broadcasting, video streaming or video on mobiles.
To test and validate a decoder implementation, reference bitstreams are provided by the BBC [bbc]. They are also used in the research area to compare and benchmark decoder implementations. For performance evaluation, this thesis uses test sequences from the **Main profile**.

### *Frame organization*

To achieve high levels of compression, motion prediction is essential and frames can be predicted from each other using motion vectors from a reference picture or frame. The notion of Group of Pictures (GOP) defines the order in which intra coded pictures and inter coded pictures are arranged. A GOP consists of a sequence of images encoded together in an encoded video stream. The types of these images follow a pattern that is repeated periodically until the end of the encoding. The visible images are generated from the coded images in a GOP. A GOP can contain the following types of images:

- **I-frame** (intra coded image) or Reference frame. It can be decoded independently from any preceding or following frame. Each GOP begins with a frame of this type.
- **P-frame** (predictive-coded picture). This frame includes difference information (motion compensated prediction) from an I-frame or a P-frame temporally prior to it. It is also a reference frame (i.e. it can be used for predicting other frames).
- **B-frame** (bidirectional predictive coded frame). A B-frame includes difference information from I-, P- or B- frames temporally located in the past or in the future within a GOP. To avoid excessive prediction error propagation, B-frames are generally not used as a reference frame.

Figure 2.4 presents a typical organization of a GOP where arrows signify "is predicted from".

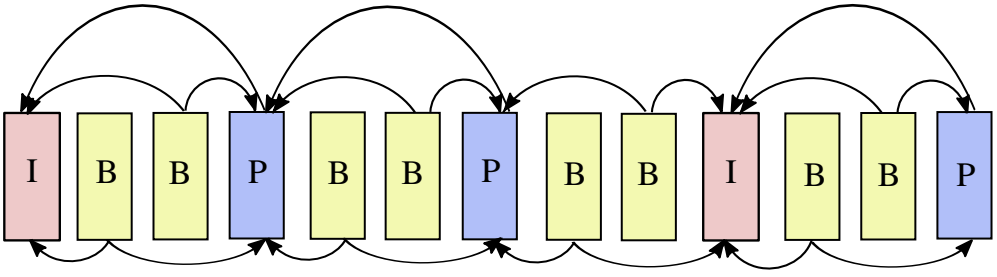


**Figure 2.4:** *Typical GOP organization*

The purpose of using different types of frames is to improve the compression ratio while containing the image distortion and providing random access to a stream. However, each type of frame requires a very different processing load and a very different amount of memory. This property is a real issue to design efficient implementations of decoders [CAMJ14, NBP+15]. Indeed, some frames can require a high processing load when others much less but they are usually all under the same real-time constraint.

## 2.3 Energy in Electronic Components

With the recent advances in the semiconductor industry, a wide range of techniques are now available to design in a low power fashion electronic systems. This section recalls the different parts of energy consumption and the techniques to reduce them.

### 2.3.1 Sources of Energy Consumption

The power consumption in System-on-Chip (SoC) is often modeled by the CMOS industry with three components as described in Figure 2.5 [Mud01].

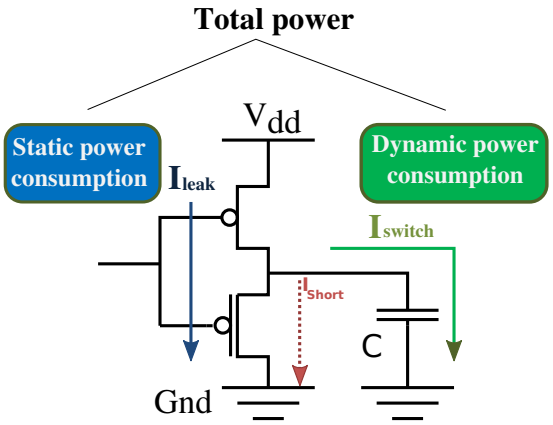


**Figure 2.5:** *Power consumption: split between the static part and the dynamic part*

They gather the essential properties for logic designers, architects and system builders. The power consumption is given by Equation (2.1).

$$P_{tot} = P_{dyn} + P_{short} + P_{stat} \tag{2.1}$$

The dynamic power $P_{dyn}$ is consumed by the charging and discharging of the capacitive load on the output of each gate. The second component $P_{short}$ captures the power resulting from a short-circuit current which momentarily flows between the supply voltage and the ground due to a short-circuit current appearing when a CMOS logic gate output switches. However this component is relatively small compared to the others [Mud01]. Hence it is neglected in the rest of the document. While the two first components are related to the circuit activity. The third component is due to the leakage current and is not related to the gate state.

### 2.3.2 Static Part

The static part of the power dissipation occurs even when a device is in standby mode. It has been neglected for a long time as it was not representing an important part of the total power dissipation. However, it becomes significant as the technology scales down. Typically, CMOS technology has been promoted for its low static power. Unfortunately, technology shrinking leads to smaller geometries which exacerbate leakage.

Static power is due to different leakage currents. Up to six sources of leakage can be considered [RMMM03] but the two main components [KAB+03] are, firstly, sub-threshold leakage and, secondly, gate leakage. Gate leakage current is due to the tunneling current

through the gate oxide insulation. As devices were scaled down, gate oxide thicknesses have decreased and an increasing probability of tunneling effect has emerged. The sub-threshold leakage current is due to a weak inversion current between the source and drain even if the transistor is OFF (subthreshold region). The sub-threshold leakage current is governed by thermodynamics, more specifically in the Boltzmann distribution [Fra06]. Equation (2.2) recalls in a compact form the static power [SNPF04].

$$P_{stat} = V_{dd}.N.I_s.e^{-V_{th}/U_t} \tag{2.2}$$

with $V_{dd}$ the supply voltage, $V_{th}$ the threshold voltage, $I_s$ the average transistor current per cell, $U_t$ the thermal voltage

Predictions of the SIA Roadmap [SIA] forecast supply voltage ($V_{dd}$) as low as 0.8 to 0.5 V in year 2018. Since a reduction of $V_{dd}$ requires an additional reduction of the transistor threshold voltage ($V_{th}$) to maintain speed, this will result in an exponential increase of the static power consumption. As a consequence, the static power on its own may already exceed the required maximum total power consumption in stand-by mode for certain applications. When the activity is very low, i.e. when very few gates are switching during a clock cycle, the static power becomes a significant contributor to the total power consumption.

Table 2.1 [SNPF04] summarizes predictions of the main parameters required for the next generations of Bulk MOSFET devices in 2018, related to High Performance (HP), Low Operating Power (LOP) and Low Standby Power (LSTP) devices. LSTP transistors have high $V_{th}$ with reduced leakage compared to other devices. Oppositely, HP transistors have the largest drive current for maximum speed at the cost of larger leakage.

| MOS Type | Physical length [nm] | Supply Voltage [V] | Threshold Voltage [V] | Ion [µA/µm] | Ioff [nA/µm] | Ion/Ioff |
|---|---|---|---|---|---|---|
| HP | 7 | 0.7 | 0.11 | 2190 | 500 | 4380 |
| LOP | 9 | 0.5 | 0.17 | 950 | 30 | 31 667 |
| LSTP | 10 | 0.8 | 0.4 | 990 | 0.1 | 9 900 000 |

**Table 2.1:** *Parameters' impact on power consumption [SNPF04]*

New design methodologies taking into account the contribution of leakage will be required for the conception of innovative SoC. State-of-art technologies (e.g. 28 nm widely available from 2015) are already largely impacted by leakage.

Similarly to previous design methodologies, solutions can be proposed at different levels, namely at circuit, architecture, and system levels. At the circuit level, many techniques have been proposed :

- Threshold voltage $V_{th}$ optimization
  - Multi $V_{th}$ technology, with fast low $V_{th}$ transistors
  - Electrical regulation of $V_{th}$ with body biasing (VTCMOS, Variable Threshold)
- Transistor optimization
  - Transistor size optimization
  - High-$\kappa$ Metal Gate
- Gated-Vdd, with $V_{dd}$ switched off in sleeping mode

This last technique is used for power-gating mechanisms. Some sophisticated mechanisms can be adopted to define a complete set of power-supply domains.

### 2.3.3 Dynamic Part

Dynamic power represents the amount of power consumed when the device is operating. It creates activity at the output of the gates when bits are changing from "0" to "1" or vice versa.

$$P_{dyn} = \alpha.C.f.V_{dd}^2 \tag{2.3}$$

where $C$ is the total capacitance seen by the outputs of the gates, $V_{dd}$ is the supply voltage, $f$ is the frequency of operation, and $\alpha$ is the processor activity (number of modified gates per clock change)

It clearly shows that, as the performance increases with the speed and the frequency, the dynamic power also increases. The dynamic power is also linked to the data activity and is in fact closely tied to the number of transistors that change state.

At the circuit level, the parameters to reduce dynamic power are the following: reduce activity when it is not necessary, reduce the capacitance due to wires between cells, and reduce the operating frequency (and the associated voltage) when possible. This latter is possible if the considered application has some intrinsic slack times.

### 2.3.4 Comparison between Dynamic and Static Parts

Some digital architectures are better than others in terms of leakage. Schuster *et al.* [SNPF04] propose a comparison of the dynamic power consumption versus the static power. Tradeoffs between dynamic and static power can be described from a theoretical point of view with the aim of understanding the influence of architectural parameters on this tradeoff and on total power consumption.

Equation (2.1) is formulated again in Equation (2.4) as the dynamic and static contributions from technological features.

$$P_{tot} = P_{dyn} + P_{stat} = \alpha.C.V_{dd}^2.f + V_{dd}.N.I_s.e^{-V_{th}/U_t} \tag{2.4}$$

with $N$ the number of cells, $\alpha$ the average cell activity, $C$ the equivalent cell capacitance, $f$ the operating frequency, $I_s$ average transistor current per cell, $U_t$ the thermal voltage.

As it can be observed in Figure 2.6, the ratio of dynamic power, $P_{dyn}$, over static power, $P_{stat}$, at the minimum of total power consumption is not constant and depends on some parameters like the activity but also on the processing frequency. In this thesis, these two parameters are adapted to the video applications to reduce system power consumption.

Subthreshold leakage is considered in static power in Figure 2.6, while gate and junction leakage are neglected. Whereas $P_{dyn}$ decreases in a quadratic manner with the reduction of $V_{dd}$, $P_{stat}$ increases instead (due to a reduced $V_{th}$). The total power presents thus a minimum i.e. an optimal power consumption (marked with a $\oplus$ symbol in Figure 2.6).

### 2.3.5 Techniques for Energy Reduction

In this thesis, the Multiprocessor SoC (MPSoC) family is targeted. This section reviews power reduction techniques in modern microprocessors. Low power modes exploit application slack times to reduce power consumption.

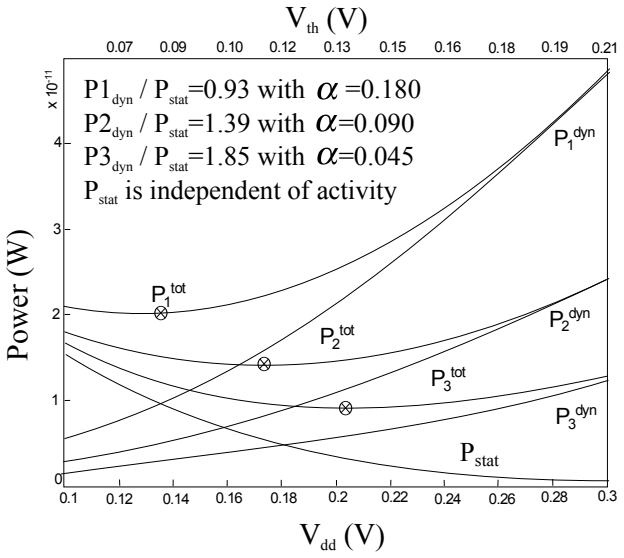


**Figure 2.6:** *Total Power ($P_{tot}$), dynamic Power ($P_{dyn}$), and static Power ($P_{stat}$) of a single square transistor, as a function of $V_{dd}$ and $V_{th}$ for three different circuit activities $\alpha$ [SNPF04]*

### Different Classes of Techniques

Figure 2.7 recalls the different architecture abstraction levels where energy saving and low power design can be taken into account. At low levels, low power techniques can be performed regardless of the application itself. These techniques were intensively discussed over the past decades [PMvKS95, RCN02].

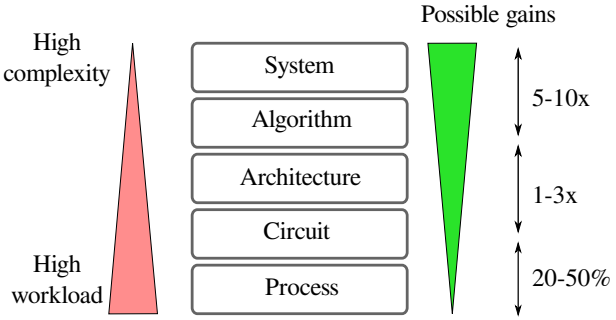


**Figure 2.7:** *Design phases versus possible energy gains*

At the lowest level, low power techniques rely on advances in the technologies and processes of the silicon industry. They require high investments.

At the next level, and among the most popular techniques, *clock gating* and *power gating* are generally implemented onto modern SoCs to provide techniques to design in a low power fashion [RTM+10]. They are independent from the application itself but are used by upper levels design techniques where an application knowledge is needed. *Clock gating* is used to turn clock tree branches off whenever they are not used [WPW00]. This technique principally reduces the dynamic power part and can be implemented at different levels of the processors from functional units to complete cores. *Power gating* targets the reduction of static power consumption by shutting-down cores during the idle phases [JMSN05]. The implementation relies on a multi-threshold CMOS process. Low $V_{th}$ transistors are used for logic blocks. They provide low latency but at the expense of a high leakage current.

High $V_{th}$ transistors are used for switching transistors. These transistors control the supply of the logic blocks and provide low leakage current.

The next class of techniques works at the architecture level. This class seeks two important goals: exploiting the parallelism and speculate on the application behavior regarding the data usage and related processing operations.

In the scope of this thesis, the considered techniques focus on the architecture level and above. Thus, the applications and their associated system requirements must be known for an efficient usage of the proposed techniques.

### Slack-time increase

Significant dynamic power reduction can be obtained by reducing the supply voltage but at the expanse of the transistor latency. Thus the supply voltage reduction must go with a clock frequency reduction. Real-time applications can benefit of frequency and voltage adaptation by increasing the slack-time. Parallelism exploitation is a good way to increase the slack-time.

**Data level parallelism** When considering an application, the primary strategy consists in optimizing the application running time, exploiting data-level parallelism. Applications that need a large amount of processing resources can use the inherent data parallelism with the Single Instruction Multiple Data (SIMD) techniques. SIMD was introduced first in the 70's and is part of the instruction set supported by the Intel x86 Complexed Instruction Set Computer (CISC) processors through Streaming SIMD Extensions (SSE) and Advanced Vector Extensions (AVX) tools. From 2009, ARM has also introduced its own vector management for the Cortex-Ax called NEON family [ARM12]. SIMD is widely used for embedded devices and is suitable for video compression and decoding [MJR+13]. It results in an acceleration of the process and hence leads to low power designs [IRF09].

SIMD operations process several data of the same type and size simultaneously. Compared to a sequential implementation, a theoretical speed-up equal to the number of data that can be handled by a single SIMD vector can be achieved. The actual speed-up depends on the number of data that can put together within one SIMD register. For example, video and graphics application handle 8-bit data types. The processor can generate multiple results of the application by packing them into a single 32-bit vector and providing SIMD vector operations. A typical utilization is depicted in Figure 2.8.

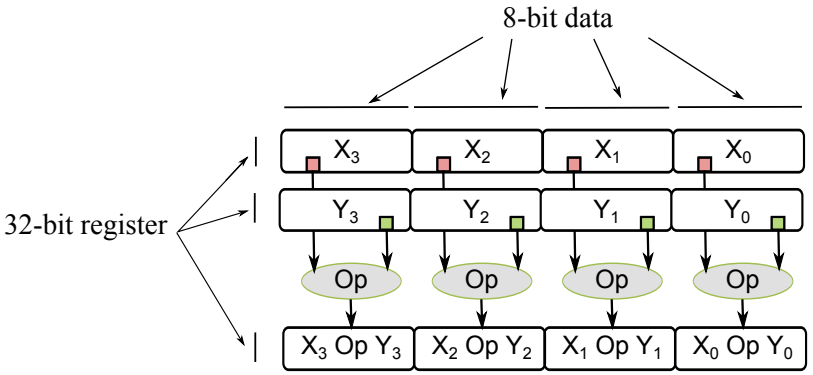


**Figure 2.8:** *SIMD vector utilization*

Using SIMD instructions can be done in three different manners:

- **compiler auto-vectorization**: all the work is left to the compiler. It shall locate and automatically vectorize the suitable loops. The performance of the compiler is then essential to achieve high gains. However, recent benchmarks [MGG+11] show that state-of-the-art compilers are capable of vectorizing only 45-71% of synthetic benchmarks and 18-30% of real application codes
- **compiler supported intrinsic functions**: intrinsic functions provide some kind of level of abstraction and can be used in conjunction with top level design tools [Cor07].
- **low-level assembly code**: writing low level assembly code is often considered to be the best approach as it consists in mapping by hand the data to the vector register. It guarantees a full utilization of the available resource. However it is time consuming and, possibly, error prone.

Raffin *et al.* [RNH+15] use SIMD on an ARM platform to speed-up the filtering process in an HEVC video decoder. The functions are coded in assembly code. Thanks to the reduction of running time of the application, the data level optimization can reach up to 40% energy reduction on a platform dedicated to mobile phones applications.

**Thread Parallelism** Once the application is optimized from the data point of view, the next step consists in parallelizing the process at a higher level. In modern MPSoC architectures, the intrinsic performance of the processor has increased without increasing clock frequencies. The same processing is split between available cores delivering the output earlier, thus reducing the running time as depicted in Figure 2.9. For example, if a perfect parallelization on two cores is assumed, then the processing can be halved with the same time of execution.

This optimization relies on the assumption that the application can be parallelized. It is typically the case for video decoders.

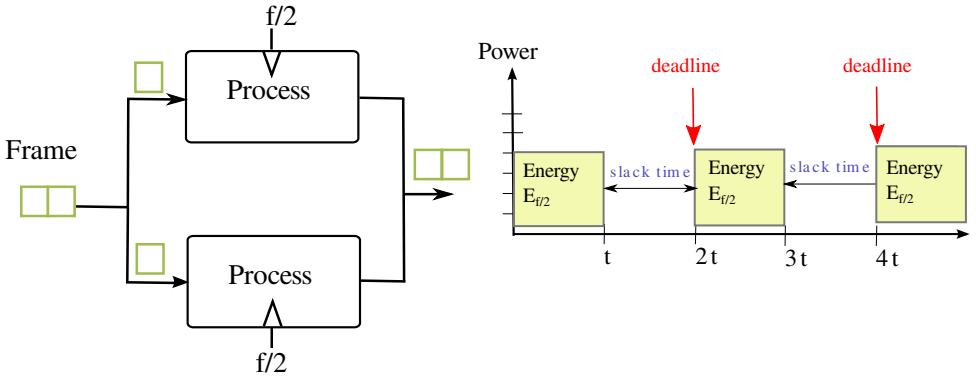


**Figure 2.9:** *Parallel thread management*

Three high level parallel processing solutions [CAMJ+12], namely independent slices, tiles and wavefront decoding, are defined in the HEVC standard to simultaneously process multiple regions of a single picture.

Frames can be partitioned into one or many independent horizontal slices, mainly to increase the bitstream robustness. The tile concept splits the slices into rectangular groups of Coding Tree Unit (CTU) called tiles. Independent slices and tiles reset the probabilities of the CABAC entropic coding and remove intra prediction dependencies and thus can

be used for parallel encoding and decoding. However, limiting intra prediction and resetting the CABAC probabilities decrease the coding performance in terms of rate distortion tradeoff, especially for large numbers of tiles or slices per frame. Moreover, in-loop filters cannot be performed in parallel at the tile or slice edges without additional control mechanisms. The Wavefront Parallel Processing (WPP) solution complements slices and tiles by splitting the frame into CTU rows [CHP11]. In the WPP mode, the CABAC context is initialized at the beginning of each CTU row. The overhead caused by this initialization is limited since the CABAC context at each CTU row is initialized by the CABAC context state at the second CTU of the previous CTU row.

Figure 2.10 illustrates the concept of the frame based parallelism where all frames within a GOP are decoded in parallel. The frame-based parallelism requires communication between threads decoding different frames in order to ensure that the Prediction Unit (PU) used as reference for inter prediction is already available (decoded) in the reference frame. Therefore, the encoding/decoding of $n$ frames can be carried out in parallel with a delay in the Inter coding configuration. The performance of the frame-based parallelism strongly depends on the coding structure and the ranges of the motion vectors. Moreover, the frame-based parallelism improves the decoding frame rate but not the decoding frame latency.

When the frame-based parallelism is enabled in *OpenHEVC*, $n$ instances of the decoder are created, with $m$ the number of decoding threads. Each decoder has its own local and global structures, while some information in the global structure are shared between the $m$ decoders, such as the Decoded Pictures Buffer (DPB) and the lists related to the video headers. An inter-thread control solution is implemented to ensure that the PU required for motion compensation is decoded at the reference frame. Otherwise, the decoding thread waits until the PU is decoded in the reference frame. Once the required PU is decoded, all waiting threads are unblocked to pursue the decoding process.

All in all, it shall be noted that HEVC is designed with a special care on parallelism. By using parallelism combined with the MPSoC capability of processing onto several cores at the same time, a low power design can be proposed as described in several studies [CAML+13a, RNH+15]. For example, Raffin *et al.* report gains up to 45% on a embedded platform [RNH+15].

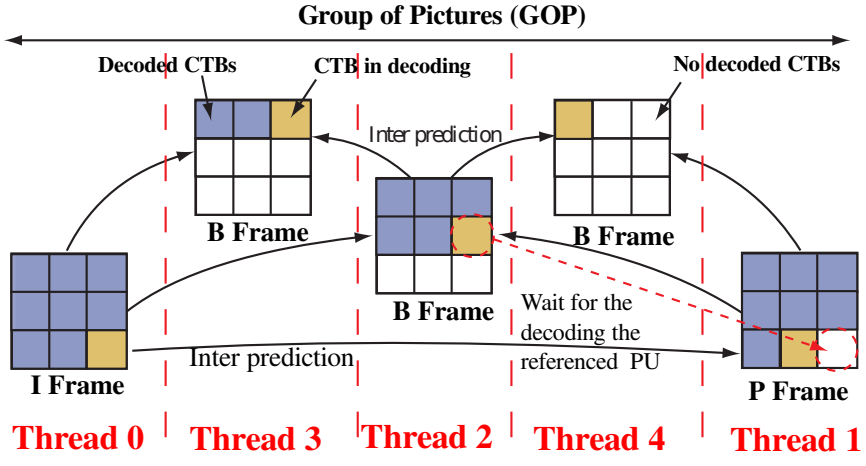


**Figure 2.10:** *Principle of the frame based parallelism in HEVC [HRD14a]*

### Dynamic Power Management

As depicted in Figure 2.11, any real-time system has several constraints on the execution itself $s_i$ and on the deadline to guarantee the system to be real-time $d_i$.

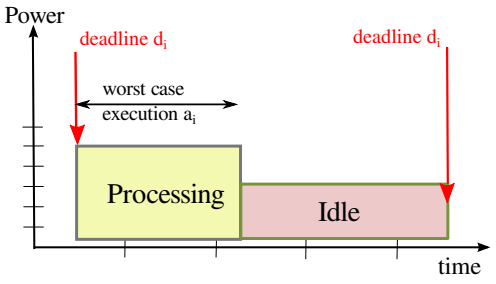


**Figure 2.11:** *Real time case*

In the case of a real-time execution with idle time (also called slack time), the overall energy is given by:

$$E_{tot} = E_{processing} + E_{idle} \tag{2.5}$$

where $E_{processing}$ and $E_{idle}$ are functions of the power consumption during the processing and idle phases. These power values depend on the supply voltage $V_{dd}$ and the processing frequency $f_{proc}$.

Applications require resources in order to perform the intended functionality, which results in a power dissipation $P$ of the CPU over a time $t$. Since the energy consumption is the integration of $P$ over $t$, an energy efficient execution is obtained when the integral is minimized. The power $P$ is further divided into the sum of the dynamic power $P_d$ and the static $P_s$, hence $P = P_d + P_s$ as seen earlier in Equation (2.4).

The popular execution strategy called *race-to-idle* [RLdS+09] was implemented to execute a task as fast as possible, after which the processor enters a sleep state (if no other tasks are available). Race-to-idle minimizes $t$, but on the other hand, it results in high power dissipation $P$ during execution. A strategy such as race-to-idle will have a negative impact on energy efficiency if the decrease in time is less than the increase in power i.e. $\Delta^- t < \Delta^+ P$ compared to running on a lower clock frequency. Depending on the CPU architecture and the manufacturing technology, this relation varies, but with current clock frequency levels, is it usually very energy inefficient to execute at high clock frequencies [ZbYAd11, RR12]. It is also inefficient to execute at very low clock frequencies [HHL+13] since the execution time becomes large and the static power is dissipated during the whole execution.

For energy efficient system design, many SoC propose Dynamic Power Management (DPM) and Dynamic Voltage Frequency Scaling (DVFS) techniques for reducing energy consumption. They can be applied at the operating system level or at the application level to address the system requirements displayed in Figure 2.11.

#### *Deep Sleep Management*

To reduce power consumption, one strategy is to manage as smartly as possible the balance between processing time and idle time. In modern processors, DPM [CCCY06] is used to reduce leakage power. The DPM technique exploits the idle period to enter into a sleep mode that consumes less energy. Figure 2.12 depicts the principle that is used. It outlines

that before entering a sleep state, a command is issued by the system. Similarly, a wake up command is also set before coming back to the active state and is managed by a Power State Machine (PSM).

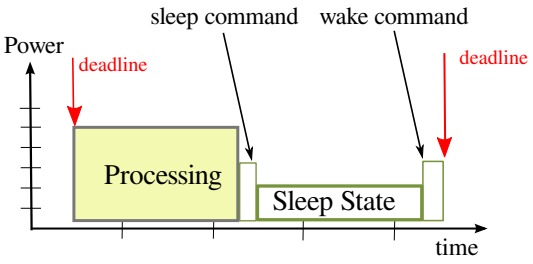


**Figure 2.12:** *DPM approach*

In [ZLC+03], each single state of the PSM is related to a manageable component mode of operation representing an instance that spans the power-performance tradeoff. It means that the power consumed can be reduced by entering a low power operating mode (sleep power state) whenever possible. State transitions have a power and a delay cost and they are triggered according to a pre-defined power management policy as depicted in Figure 2.13. In general, low-power states have lower performance and larger transition latency when compared to higher power states. Because of this, the process of shifting between modes must be coordinated, since the power saved in the low power state must compensate the power and time spent in the transition.

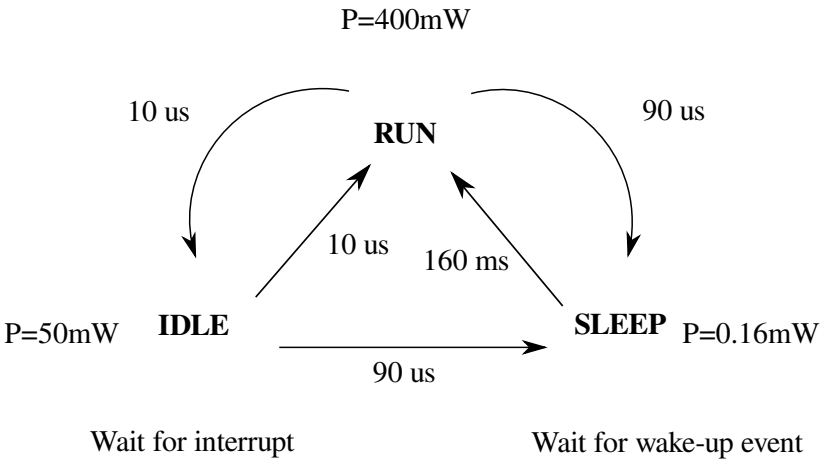


**Figure 2.13:** *Power state for StrongARM-110 [ZLC+03]*

Figure 2.13 is the PSM model of the StrongARM AS-1100 [ZLC+03]. Each state in the machine is marked with power dissipation and performance values, and edges are marked with transition times. Notice that both IDLE and SLEEP states have null performance, but the time spent to exit the SLEEP state is much higher than to exit the IDLE state. However, the power consumed by the chip in SLEEP state is much smaller than in IDLE because of the static power consumption of Equation (2.4).

A low power design using this technique aims at finding the best tradeoff of low power modes and transitions in the PSM. The two-state case in which there is a single active and a single sleep state is a continuous version of the ski-rental problem. If the idle period is long, it is advantageous to transition to the low cost state immediately; if the idle period is

short, it is better to stay in the high-cost state. An online algorithm which does not know the length of the idle period must balance these two possibilities.

This design item is a not trivial task. Augustine *et al.* [JAS08] compare it to the *multislope ski rental problem* as depicted in Figure 2.14. It shows the total energy consumed as a function of the length of the idle period. They state in their work that to reach optimality the system shall know the idle time period. It can be either determined off-line or using probability-based strategy. In practical implementations, estimating the length of an idle period can be based on recent history [PLB07].

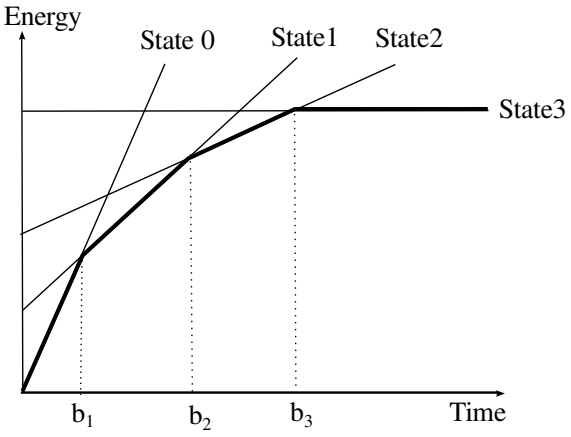


**Figure 2.14:** *Energy consumed by the optimal strategy as a function of idle period length*

Coming back to Equation (2.5), the resulting power of a DPM enabled-system is:

$$P_{dpm} = P_{processing} + P_{sleepstate} \tag{2.6}$$

where $P_{processing}$ is the same as in Equation (2.5) and $P_{sleepstate}$ is power of the associated sleep state with the addition of the needed power for the sleep command and the wake up. For real-time systems that are designed under a deadline management, monitoring the processing time with respect to the deadline requirement, leads to choose the appropriate sleep state.

### *Dynamic Frequency Scaling*

The other mean to reduce power from the situation described in Figure 2.11 is to slow down the working frequency of the processor. From Equation (2.4), the dynamic power can be drastically reduced. DVFS was introduced in the 90's [Ogu90] offering great promises to dramatically reduce power consumption in large scale digital systems. It consists in adapting both voltage and frequency of the system with respect to the changing workload. Most of the legacy SoC provide tables to control frequency and voltage at the same time. An example is given in Table 2.2 for the Intel XScale PXA27x.

The optimization methodology consists in finding the best slowing down factor. Figure 2.15 shows an example of a speed-up strategy.

Lowering down the frequency induces a tradeoff with the performance of the system. But in real-time scenarios, high performance is not needed all the time. It is especially the case for video decoding where a steady playback is demanded (e.g. 25 frames per second) for the whole execution. The actual performance of the application could be significantly higher and the performance should be adapted over time to reduce power consumption.

| Index | Dynamic Power (mW) | Frequency (MHz) | Voltage (V) |
|---|---|---|---|
| 0 | 925 | 624 | 1.55 |
| 1 | 747 | 520 | 1.45 |
| 2 | 570 | 416 | 1.35 |
| 3 | 390 | 312 | 1.25 |
| 4 | 279 | 208 | 1.15 |
| 5 | 116 | 104 | 0.9 |

**Table 2.2:** *Intel XScale PXA27x power characteristics*

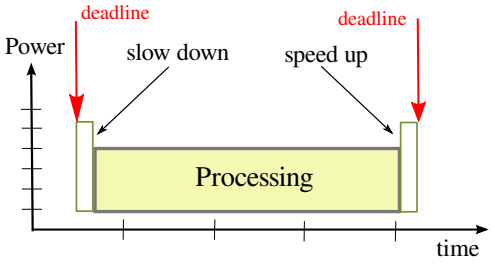


**Figure 2.15:** *Dynamic Frequency Scaling approach*

### Operating System Power Management

Choosing the power states is mainly performed by the operating system. This section provides an overview of how the Linux operating system controls these low power states.

#### *Advanced Configuration and Power Interface (ACPI)*

The Advanced Configuration and Power Interface (ACPI) standard [Bro05] specifies power states for devices. The state of a device is either working, stand-by or hibernating and noted as its *global state*. When in working mode, the processor can be in different states noted as C-states. It is then active. Table 2.3 summarizes the most common states. More power saving actions are taken for numerically higher C-states. Lower power C-states have longer entry and exit latencies

| C-state | Description |
|---|---|
| C0 | Processor is active |
| C1 | Processor is halted |
| C2 | Processor is halted and most clocks are stopped |
| C3 | Cache is flushed - Deep Sleep |
| .. | .. |
| C6 | Deep power down |

**Table 2.3:** *C-states description*

When the processor is in active state C0, the cores can use different processing frequencies using the DVFS mechanism. In the ACPI standard it is refered to as P-states. P0 attributes the highest available performance point (i.e. highest processing frequency

and supply voltage in the pool) while Pn attributes the lowest performance point. Modern processors enable a wide range of frequencies offering many DVFS points.

The operation system is in charge of controlling both P-states and C-states. They are accessible in Linux through the *cpuidle* driver (C-state) and *cpufreq* driver (P-state).

#### *Cpuidle for sleep state control*

The *cpuidle* driver [PLB07] controls the C-states usage. Two governors, *menu* and *ladder*, are available to manage the transitions transparently. To choose the most appropriate state to enter, a tradeoff needs to be found. It takes into account the energy cost, the sleep command latency and the wake up command latency. These parameters are tightly coupled with processor characteristics (Intel or ARM for instance). Therefore vendors usually provide their own optimized governors (e.g. *intel_ idle*, *exynos_ idle*, ...).

#### *Cpufreq for dynamic frequency scaling control*

The *cpufreq* driver controls the P-states usage. Associated to this driver, several governors are available: *powersave*, *conservative*, *ondemand*, *performance* and *userspace*. While *performance* and *powersave* set the current state respectively to P0 and Pn, the *ondemand* and *conservative* governors use a dynamic strategy based on the actual load of the processor. Finally, the *userspace* governor leaves the user to set the processing frequency as desired. This strategy is widely used to embed within the application smart dynamic frequencies strategies. Frequency changes can easily be done with system calls within an application as described in listing 2.1.

```
#include <cpufreq.h>
application(){
        cpufreq_set_frequency(Core0, FREQ1);
        }
```

**Listing 2.1:** *Userspace cpufreq call*

#### Discussion

In this section, various techniques have been presented that reduce the power consumption of embedded systems.

We can notice that these techniques are well suited for the video applications presented in Section 2.2. For example, data parallelism can be used efficiently in the filters that are embedded in video decoders. Task parallelism can be used to process slice, tiles or frames in parallel. Frequency scaling can be used to adapt the processing frequency to the varying demand of processing per frame type. The next section presents the main features that can be used to evaluate the performance of the obtained system.

## 2.4 Performance and Quality Assessment

In the previous sections, different means to design low power systems are described. However they come generally together with a reduction of the overall performance of the system. Focusing on the video processing case, this section outlines how the Quality of Service (QoS) can be assessed in order to evaluate methods for low power system design.

### 2.4.1 Video Decoding Set-up

Figure 2.16 describes the general set-up of a video decoder. Generally, the quality of the input bitstream is measured with the frame loss and the bit error rate. These metrics are affected by the transmission and transport parts. Once this part is secured, further parameters can affect the overall QoS and Quality of Experience (QoE) from the video decoding element to the rendering process [WSV+05]. These parameters are the delivery performance before a deadline, latency and the power budget.

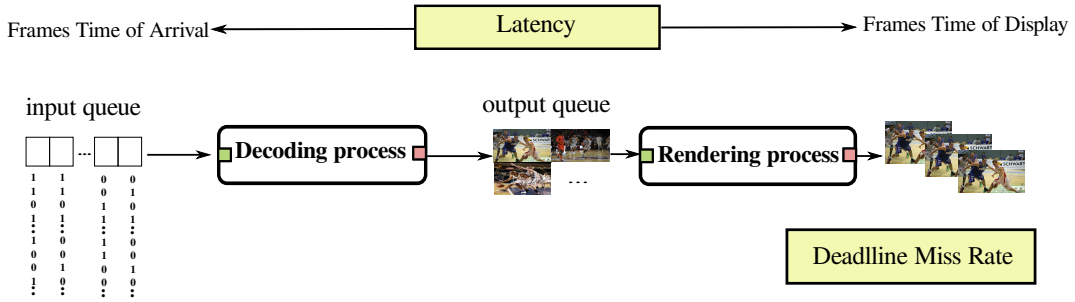


**Figure 2.16:** *Basic video decoder set-up*

From the user point of view, two different cases must be considered. Either the user demands a real-time video rendering or the user only requires off-line decoding. The former case imposes deadlines to the system. Indeed the video shall be displayed at a defined rate fixed at a system level. Most of the modern use cases rely on this: broadcast TV, streaming, video conference calls for examples.

In the latter case, only a decompression of the video is required and real-time rendering is not demanded. The deadline requirement for the decoding to be finished is then loose. The typical applications are VOD or video transcoding (e.g. H.264/AVC to HEVC).

In both cases, the latency performance can be critical but depends on the application. Video conferencing applications can be cited as critical in terms of latency between receiving and displaying the video.

### 2.4.2 Deadline Miss Rate (DMR)

For the real time video decoding use case, the Dead Line Miss Ratio (DMR) is a prime performance criterion. The system is set to render, display or show the input sequence at a given frames per second (fps) for the playback. Each frame of the video is bound to be shown before a deadline, otherwise a deadline miss is hit. To be on the safe side, many system designers decide to process the video as-fast-as possible to respect this requirement. Chapter 4 and Chapter 5 analyze this strategy and show that better management techniques can be proposed.

### 2.4.3 Latency

To overcome real time issues, typical video decoder set-ups use lag-time buffers that store several frames in advance of the display deadline. However, applications such as live TV or video conferencing have high requirements on the overall latency induced by the decoder. In this latter case, only a minimal number of frames can be used for lag-time management. Chapter 5 analyzes latency performance when coordinated frequency schemes are implemented.

### 2.4.4 Power and Energy

The term *power* is often used to refer to what is really energy. In portable systems for instance, the amount of energy needed to perform a task is the quantity to monitor, as portable systems are supplied by a battery of limited energy budget. Batteries store a fixed amount of *energy* and not *power*.

The *power* consumption $P_{tot}$ in CMOS electronics is defined in Section 2.3 with Equation (2.4). The *energy* consumption $E$ is the amount of *power* supplied to the application in a period of time $T$ as given by Equation (2.7).

$$E(T) = \int_0^T P_{tot}(t)\,\mathrm{d}t \tag{2.7}$$

From Equation (2.7), it shall be noted that evaluating the *energy* performance of a system suggests to access the instant *power* during the evaluated period $T$. Therefore, accurate energy measurements need a small enough sampling period of the instantaneous *power* of the *under-test* period. In this thesis, we mainly use power sensors called INA231 where data are collected at a sampling rate of 10 Hz through I2C bus. Even though the sampling time can considered as small for video applications, short variations can be monitored. The sensor averages out the power consumption as described in the datasheet of the component [Ins]. Moreover, our experiments are repeated several times and averaged to obtain accurate power estimations. As highlighted in recent studies [NHP+14, CAMJ14, RNH+15], this indicator can be used of energy monitoring with accuracy to compare design strategies.

Furthermore, one can note that *power* efficiency and *energy* efficiency performance are slightly different performance metrics. Designing *energy* efficient systems means increasing the stand-up time of the device. For a given QoS, systems that consume less energy are considered as better performers.

Designing a low power system is a slightly different goal. For example, within the scope of system design or fast-prototyping, one can look for the smallest frequency that can guarantee the system QoS. The smallest frequency may not be the frequency that also minimizes the energy as shown by the contributions chapters. In other applications, low power design refers to limiting the *peak power*. If it exceeds a given limit, this peak power leads to system damage.

## 2.5 Conclusion

As depicted in Figure 2.7, There exists a wide range of techniques to reduce power consumption of systems. These techniques may apply at different steps of the design depending on the freedom left to the designer. Applying low power techniques requires a deep analysis of the application characteristics. This analysis can be of two kinds:

The first type of analysis is the processing part analysis and could be called *optimized processing*. Indeed, data parallelism shall be exploited when present in the application. Processing tasks in parallel is possible if the input data are split accordingly. Dynamic power management techniques can then be used together with a smart management of the idle periods

The second type of analysis shifts the energy consumption paradigm to the top level system requirements. It could be called *adequate processing*. Once the first required step is performed completely and more savings are demanded, one can propose to analyze the QoS of the application. Reducing the QoS of the application can lead to more energy

savings. In this latter case, a tradeoff can only be done when a complete knowledge of the applications requirements is available.

In the next chapter, more insights of the video use case are given in relation with the techniques proposed in this section.

CHAPTER 3

# Low Power decoders

## 3.1 Introduction

Any system design begins with a feasibility study. It focuses first on studying its capability to fulfil system requirements. For real-time systems, the first step is to complete the processing within the allocated time budget. The next step forward is to improve the design with low power capability. For many applications that target handheld devices, it is also the fast track to early adoption by the end customer. It is particularly the case for video decoding and more precisely for the new video compression standard HEVC.

In the previous chapter, various low power techniques were introduced. They can be applied at different phases of the design and they are not particularly related to video decoding purposes. An efficient implementation needs to take into account the application characteristics and to tailor them to the processing resources. In this chapter, the related work on low power decoders is presented with respect to the different steps described in the previous chapter.

The system level approaches are presented in Section 3.2. The leading motivation is to reduce the processing load by processing less data inducing QoS or QoE reduction. In Section 3.3, another trend is presented relying on the optimization of the computation sequences by dedicated hardware implementations. Section 3.4 describes the software based implementation. In this latter case, dynamic power management techniques presented in the previous chapter can enhance the energy efficiency. Finally, Section 3.5 is an introduction of a new trend based on a tight connection between the MPEG standard and the video decoding system. It is called Green Metatada and aims to improve the energy efficiency of the decoder.

## 3.2 System Level Approaches to Low Power Decoding

The primary approach for designing a low power decoder consists in scaling up and down the system performance at runtime, performing more or less computation to mobilize more or less computing resources. From a video standard point of view, there are several types of scalability that are available. Scalability has an impact on the characteristics of the video.

| | Power results | |
|---|---|---|
| | High Resolution | Low Resolution |
| Intel-based platform | 19 W | 9 W |
| ASIC | 104 mW | 26 mW |

**Table 3.1:** *Power consumption comparison between two resolutions. High resolution is 2K for Intel-based platform and 4K for ASIC and Low resolution is 1080p for Intel-based platform and ASIC*

### 3.2.1 Changing Resolution

The resolution represents the size in pixels of each frame in a video sequence. Examples of typical resolutions are given in Figure 3.1.

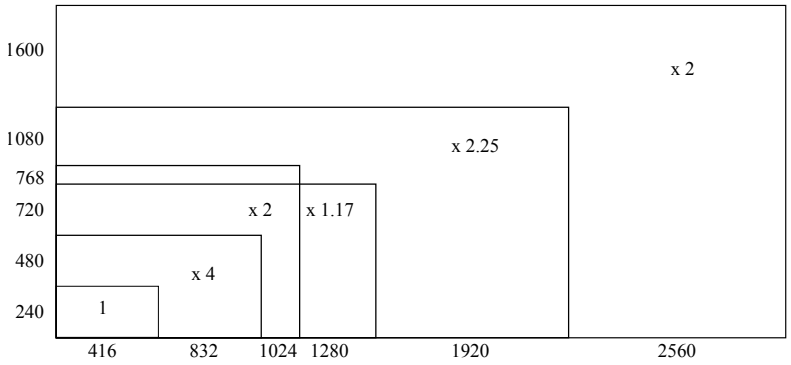


**Figure 3.1:** *Examples of resolutions*

By scaling down image resolution, one can reduce the number of operations needed to decode a video frame. Thus, the energy needed to decode the sequence is linked to its resolution.

Several studies [RNH+15, CAMJ14, LCW+15] analyze the energy consumption for different resolutions. An excerpt is presented in Table 3.1. From a system point of view, the pair encoder/decoder can use resolution downscaling to reduce to power consumption by adapting the resolution to the actual capabilities of the end devices.

### 3.2.2 Changing Sampling Time

Another characteristic of a video is its capturing frame rate. It quantifies the number of images per second of capture. A high frame rates induces better fluidity. It is especially needed for fast varying sequences like sport scenes. If the quality can be degraded, reducing the playback rate is another mean to reduce power consumption. However the expected savings depend on the type of encoding. Indeed, each frame has a similar complexity of decoding, then power savings can be achieved by skipping frames. Examples of power reduction by reducing the playback rate are provided on a typical platform for embedded systems in [NBP+15].

Figure 3.2 shows that if a dedicated power management is used such as DVFS, the gains in power may be large. This effect is due to the quadratic relation between the power consumption and the voltage/frequency values.

However, these gains are much smaller when the decoded frames have very different complexities in decoding. It is the case with the temporal scalability in Scalable HEVC

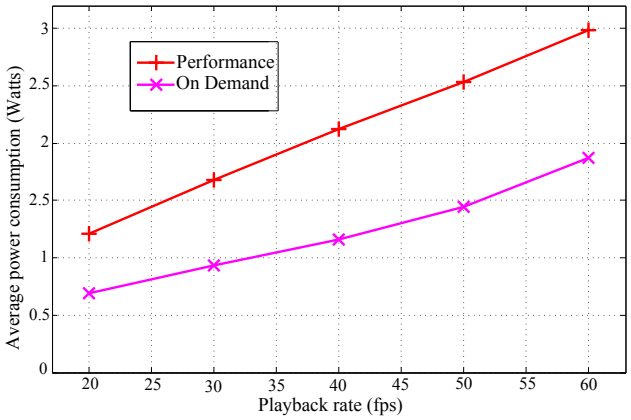


**Figure 3.2:** *Power consumption as a function of the playback rate[NBP+15]*

(SHVC). An example of 4 hierarchy stages encoding is depicted in Figure 3.3. These stages are considered as sub-layers, thus it is possible to choose the number of layers to decode. For example, if layer 1 is selected for decoding, all frames of lower layers (layer 1 and layer 0) are decoded which represents 2 frames over 8 [SOHW12b]. But the reduction in decoding complexity is not with the same order of magnitude. Indeed, predicted frames (e.g. B-frames) are less complex to decode [NBP+15]. Therefore a reduction by a factor of 2 display rate would not lead to half the power consumption but only several %.

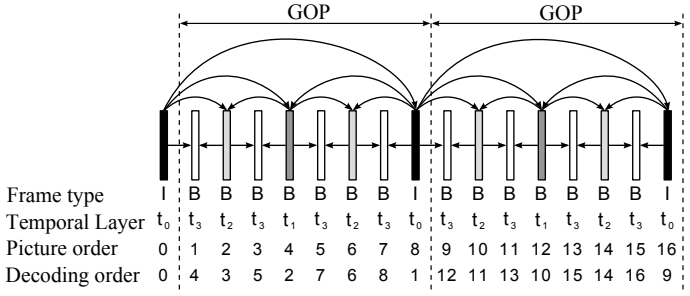


**Figure 3.3:** *Hierarchical temporal structure with GOP is equal to 8 and 4 temporal levels: from lowest level $t_0$ (including pictures 0 , 8 and 16) to the highest $t_3$*

### 3.2.3 Changing Quality

In addition to the resolution and time scalability types, modern compression standards use a Quantization Parameter (QP) to adapt the bitrate of the compressed sequence [WSJ+03]. At a frame level, the residuals of an image prediction are converted into the spatial frequency domain with a DCT. This transform is approximated into the integer domain. The QP defines the step size for converting the transformed coefficients to a finite set of steps. High values of QP mean large steps and poor accuracy whereas small QP values approximate with a better accuracy the frequency spectrum. More bits are then needed for the representation of the video, hence increasing the bitrate. Each unit of QP increases the step size by 12% and decreases the bitrate by 12% in average.

From a decoding point of view, a higher QP leads to less complexity and less power consumption. Video decoding performance analysis is traditionally performed with different QP values and confirms the capability of the QP parameter to scale down the power

| | QP 22 | QP 27 | QP32 | QP27 |
|---|---|---|---|---|
| Intel-based | 18.83 W | 16.28 W | 14.91 W | 13.91 W |
| ARM-based | 4.168 W | 3.773 W | 3.351 W | 3.073 W |

**Table 3.2:** *Power consumption (W) comparison for different QP values.*

consumption. Table 3.2 reports HEVC decoding power measurements for different QP values. It is measured on an Intel-based platform and an ARM-based platform with a 1080p sequence as input [NHP+14].

### 3.2.4 Scalable Extensions of Video Compression Standards

Standardization bodies like MPEG proposed to gather the previously cited scalabilities into standards. For H.264, the scalable extension is called Scalable Video Coding (SVC) [SMW07]. Similarly, HEVC is also extended with a scalable version called SHVC.

The purpose of these scalable schemes is to propose to the decoder a base layer including a video sequence at a minimum quality and enhanced layers with higher qualities. The enhancement can be either a higher resolution (*Spatial scalability*), higher frame rate (*Temporal scalability*) or better quality (*SNR scalability*).

Using scalability can be beneficial from the network point of view as the bitrate can be reduced compared to the equivalent simulcast scenario (sending all the streams of different qualities in the bitstream). However, from the decoder point of view, scalability leads to a great increase of the decoding complexity. In [HRD14c], a comparison is performed between an SHVC decoder and an HEVC decoder. The SHVC decoder is reported to be twice more complex when compared to an HEVC decoder on a standard SW decoder implementations.

Due to this increase of decoding complexity, SHVC is not suitable to design low power decoders.

## 3.3 The Case of Specialized Circuits for Low Power Decoding

Video processing is a complicated task for a processing unit. Indeed, it shall handle a stream of data and perform complex signal processing functions. On top of that, applications like video play-back require a QoS in terms video delivery. Therefore the trend is to first develop dedicated circuits to speed-up the intensive processing part of the system. Two approaches are widely use by SoC designers. One can develop a fully customized hardware solution called Video Processing Unit (VPU). Another trend is to use off-the-shelf Digital Signal Processor (DSP) architectures to speed-up signal processing parts of the system. Obviously, the cost and effort of these two approaches are very different and depends on the final use case. For a architecture point of view, the DSP solutions could be also categorized as a software-based solution.

### 3.3.1 Application Specific Integrated Circuit

From H.264/AVC to HEVC, several SoC makers provided implementations of hardware decoders [ZZH+11, LCW+15]. The challenges are then to minimize the number of gates, silicon size and power consumption. In return, hardware implementations are costly, demand high engineering resources and are not future proof if the standard evolves.

Raffin *et al.* [RNH+15] compare the power consumption of HEVC circuits with other implementations based on software. They show in particular that Application Specific Integrated Circuit (ASIC) decoders surpass all software ones. For example, the energy consumption per pixel is compared between ASIC-based decoders and software-based decoders. For 4K video, the ASIC decoder consumes less than 1 nJ per pixel when the proposed software-based decoder consumes 20 nJ per pixel on a 1080p sequence. However the gap between the implementations shall be qualified. Indeed the hardware results focus on the chip consumption only. Peripheral items like memory access are not accounted in the power measurement.

Benmoussa *et al.* [Ben15] compare the power consumption of VPU architecture with General Purpose Processor (GPP) solutions. Hardware solutions can reduce the power consumption by a factor from 2 to 4. Indeed the gap between hardware solutions and software solutions can be bridged thanks to parallel processing on multi-core architectures.

On top of requiring high workload at the development phase, the main drawback of solutions based on hardware specialized circuits is their total or partial incapacity to support major changes. They are often considered as *not* future proof when the technology turnover is less than a few years.

### 3.3.2 Digital Signal Processors - DSP

Within the class of processors, DSP are suited to signal processing applications. They provide a specific set of instructions that improve the efficiency of resource intensive parts of the system. Because video decoding is mostly composed of intensive signal processing functions, using DSP cores seems to be a natural way to implement energy efficient decoders.

Benmoussa *et al.* [BBSB13] benchmark H.264/AVC video decoding on DSP. They compare it to more generic processors. The authors use an OMAP3530 that implements the typical DVFS and DPM mechanisms. The SoC also embeds a generic ARM processor and is based on a 65 nm technology. The conclusion is that performance is quite similar for low resolutions but as the resolution grows, the ARM power consumption increases unlike the DSP power consumption. However, it shall be noted that DSP implementations imply a specific description or engineering of the reference source code leading to more development efforts.

## 3.4 The Case of Software-based Video Decoding Based on a General Purpose Processor

The performance of modern GPP platforms is continuously increasing on several aspects which include processing frequency range and number of supported cores. GPPs also address different segments from offline processing to embedded processing. In HEVC, a special care was taken to minimize the decoder complexity and to make the decoding procedure parallel by nature.

### 3.4.1 Intel-based implementations

There exists a wide range on Intel-based platforms. More specifically, several studies proposed efficient implementations supporting real-time decoding for HEVC [BBSF12, HRD14c, BAMG+13]. The initial goal of these studies is to prove the feasibility of pure

software decoding implementations. These implementations rely on the following optimization methods.

- Streaming SIMD Extensions (SSE) are used to accelerate intensive signal processing computations
- Parallel computing is used to process frames in parallel and to use the intrinsic speed-up due to parallel processing. Figure 3.4 exposes the speed-up obtained with the parallel processing.

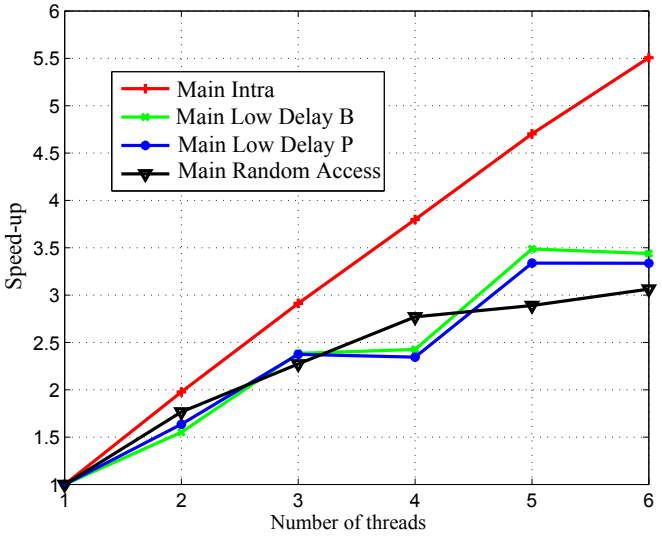


**Figure 3.4:** *Speedup performance of the frame-based parallelism implementation decoder run on a general purpose processor [RNH+15]*

Recent works [BBSF12, BAMG+13, DSY+14, HRD14c] show that such implementations can support high resolutions such as 4K and 8K, challenging then dedicated hardware platforms.

### 3.4.2 ARM-based implementations

Compared to Intel-based platforms, the objective here is to address slightly different applications where mobility is essential. ARM-based implementations aim at providing software decoders for embedded systems. These systems are likely to be battery-supplied, hence their power consumption is a primary concern.

Similarly to Intel-based platforms, ARM-based platforms benefit from efficient multi-core architectures design and large ranges of processing frequencies. Hence, several studies demonstrate software decoder implementations on ARM-based platform [CAMJ14, DSY+14, RNH+15]. These implementations use the same optimization methods as for Intel-based platforms.

- SIMD optimizations with the dedicated NEON instruction set [ARM12].
- Parallel computation is used to process frame in parallel with the same management exposed for Intel-based platforms

To address energy efficiency purposes, ARM launched a dedicated architecture called big.LITTLE where a cluster of powerfull (big) cores is combined with a cluster of energy efficient (LITTLE) cores. Several vendors provide MPSoC based on this architecture. The Exynos 5 is one of them and is commonly used in recent devices [exy13].

### 3.4.3 Software-based Implementations

Several implementations exist for different purposes. The most famous one is the reference software used by standardization bodies [Bos12]. This implementation does not target real time applications and cannot be used for power and energy evaluations. Several other implementations are developed in a closed-source manner and are not available easily.

Finally, the *OpenHEVC* solution [Ope] is used throughout this thesis as it is an open source implementation and can be modified for research purposes.

## 3.5 The Normative Approach: MPEG Green Metadata

### 3.5.1 Scope and Requirements

Conscious of the greater need to provide techniques that could save power on video services, MPEG issued a call for proposals (CFP) on energy-efficient video processing in April 2013. The responses show that utilization of metadata is an efficient way to reduce the power consumption of handheld devices. It mainly focuses on the decoding part. Therefore MPEG initiated the standardization of Green Metadata [MPE]. The system architecture is described in Figure 3.5 [FDM+15].

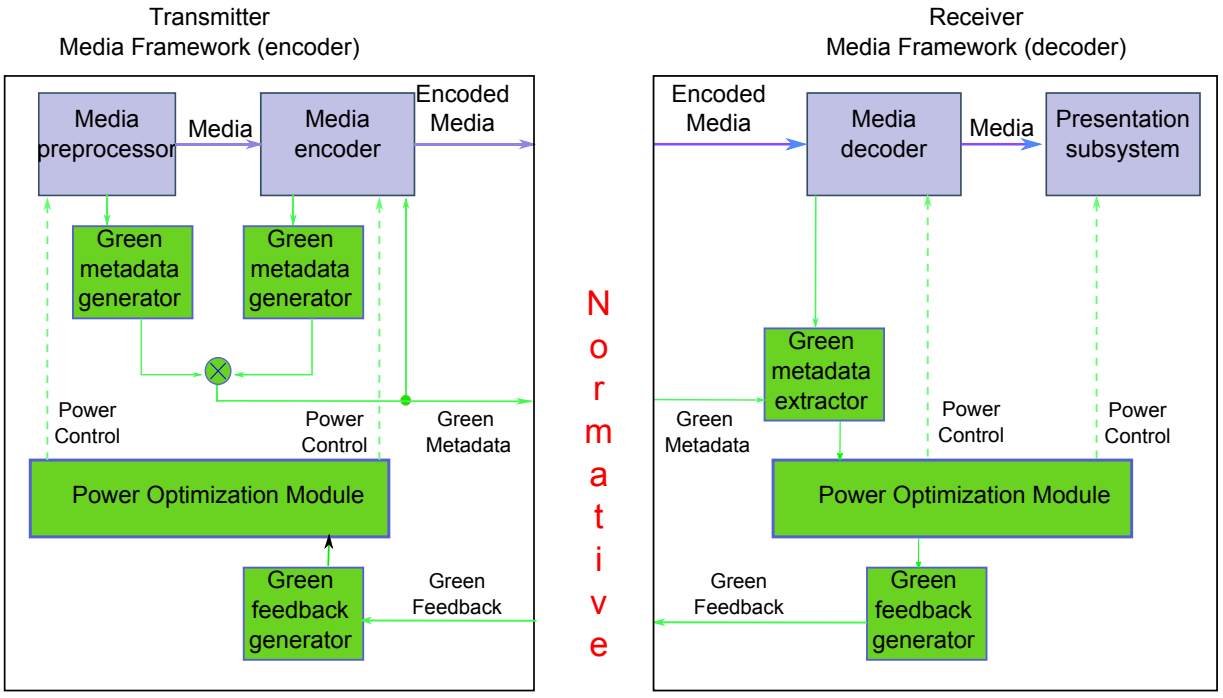


**Figure 3.5:** *Functional architecture based on Green Metadata [FDM+15]*

Four proposals are being standardized in the first version of Green Metadata and are described hereafter.

### 3.5.2 Current Green Metadata Proposals

#### Display-Adaptation metrics for Display Power reduction

This technology was submitted by Samsung [Ele]. To quantitatively understand the power consumption of the recent mobile device, Samsung measured the component power consumption on a smart phone. In the measurements, a streaming use case is used as the video application and WiFi is used as the communication method. The results show that 300 mW are needed for downloading the video, 300 mW for decoding it, 400 mW for displaying it, and 250 mW are spent in idle mode. From a circuit and hardware point of view, reducing the supply voltage is an efficient technique to reduce power consumption. Therefore, the proposal relies on lowering the display socket of the device as proposed by Shin *et al.* [SKCP11].

Figure 3.6 shows an illustration on display power reduction based on display adaptation. As a normal display solution used for current mobile devices, video frames or images are stored in a frame-buffer and fetched to be rendered on the display screen, with a normal backlight (for LCD) or voltage level (for OLED), as shown in the top part of Figure 3.6. However, the Red Green Blue (RGB) values in the frame-buffer can be scaled up while reducing accordingly the backlight or voltage [SKCP11]. While the power is reduced, the inherent quality degradation can be partially retrieved with the adaptation of the pixels properties.

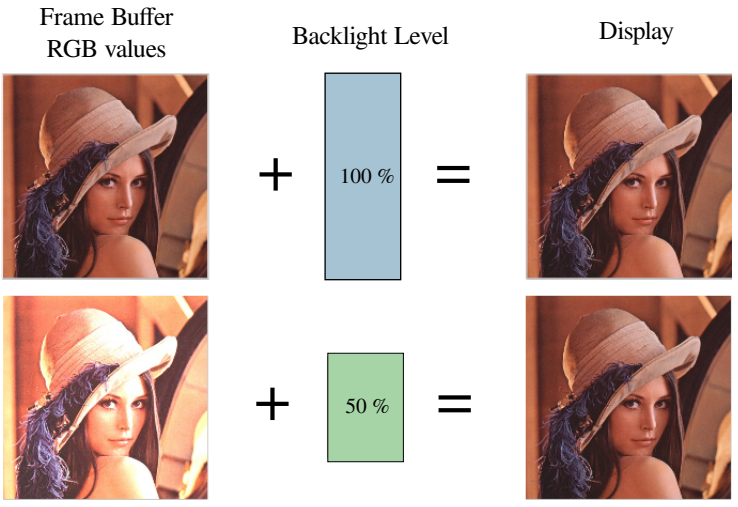


**Figure 3.6:** *Display Adaptations for Power Reduction [Ele]*

To enhance this technology, the information about the pixel distribution of each video frame needs to be provided to the power control module. It is done with a metadata. Both OLED and LCD displays can reduce their power consumption if the content distribution parameters are known. This parameter is the maximum pixel value in a frame. It is denoted as $X_{max}$. If each pixel is scaled up by a factor $255/X_{max}$, then the image rendering will brighter than the original one. To compensate this increase of brightness, the original backlight is decreased accordingly from level $B$ to $B * X_{max}/255$.

The loss of quality is then controlled and gains in energy up to 30% can be obtained [Ele]. Within the scope of Green Metadata in MPEG, the encoder embeds the $X_{max}$ value in the bitstream and this embedded metadata allows the display unit to scale down the LCD

backlight or the OLED voltage appropriately. Depending on the application, the feature can be implemented at different levels GOP-based, scene-based or time interval-based.

To provide higher gains, the decoder can go further into pixel intensity upscaling. In this case, some pixel reach saturation level leading to a contrast degradation [FDM+15]. The induced quality distortion shall be processed especially as proposed by Cheng *et al.* [CHP04]. By using a metadata, the encoder can provide information on pixel distribution for a given time sequence and help reduce distortion.

### Decoder power reduction

For processing application, the power consumption of CMOS systems is dominated by dynamic power as specified in Equation (2.4) when the clock frequency increases. DVFS techniques are widely used in modern implementations to reduce the dynamic power as presented in Section 3.4. However, the main challenge remains in adapting the processing frequency to the actual needs of the processor to decode video frames. The complexity of video frames is varying a lot from frame to frame and highly depends on compression rates. Therefore, it is not possible for the decoder to choose in advance the adequate processing frequency. This effect seriously affects the energy efficiency of video decoders [HLC+05].

A Green Metadata proposal is based on a complexity estimation of the decoded stream from a metadata generated by the encoder. An efficient protocol for complexity signaling consists in specifying the period over which complexity is constant. The standard proposes several period-types to which the complexity information applies.

Furthermore, the encoder has no information on the decoder hardware architecture. Therefore, the complexity metadata is expressed on a module-based approach. Six different indicators represent the overall complexity: entropy decoding, dequantization and inverse transform, intra prediction, motion compensation, deblocking, and the rest. Each indicator is represented by its complexity in cycles and can then be used by the decoder to compute the expected time of processing. The decoder can then stretch the processing adequately to meet the requirements fixed by the play-back rate. It shall be noted that the current proposal only addresses the H.264 standard. HEVC has not been considered yet.

The authors of this metadata present results in [ea13] with a comparison versus the *Performance* and *Ondemand* governors provided by typical Linux distributions. Gains up to 12.5% on power are reported in average.

The main drawback with this proposal is the possible mismatch between the complexity estimation computed by the encoder and the actual capability to decode of the decoder. This proposal can be efficient only if the decoder can exploit correctly the information contained in the metadata. A training period is needed to fit the metadata into a model of execution but this process is not provided in the current technical proposals.

### Green Adaptive Streaming

Video over Internet is the main contributor of the exponential growth of data exchanges. Streaming services ease the deployment of new applications. They can use network flexibility and media can be adapted dynamically. Exploiting the inherent scalability of video contents, MPEG Dynamic Adaptive Streaming over HTTP (DASH), standardized in 2001, addresses this field [Sod11]. Its principle is based on a client-side selection of the video representation based on network capability. The input bandwidth is monitored and the client can select an appropriate representation that can fit into the budged that is offered by the network. On the server side, several levels of representation are made available to

the client. They are divided into time segments. A segment can hold 1 second of media for example.

Green Metadata adds a new dimension into this system framework. Indeed, the media selection on the client side can also be adapted to the power consumption of the decoder. Figure 3.7 shows the block diagram of the DASH that supports green applications.

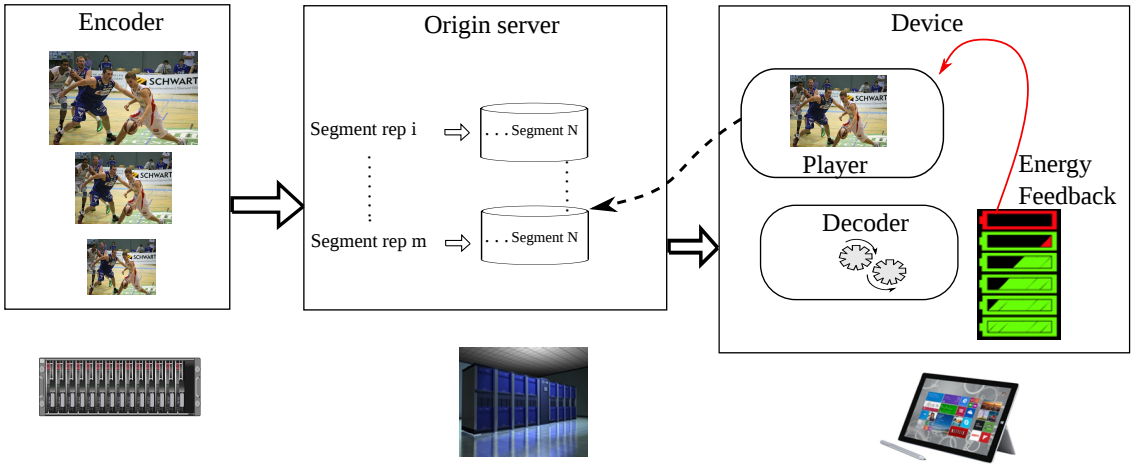


**Figure 3.7:** *Adaptive streaming proposed by Thomson Video Networks*

If the power resource is constrained, three strategies are possible to select the correct media. Firstly, the device chooses a media segment requiring low power consumption. Obviously it induces a reduction of QoE because of lower resolution and stronger quantization. Secondly, the device can measure the actual power consumption of the current segment and then react accordingly for the next media segment. The device is always handling the power consumption with a delay of one-segment. Thirdly, the device uses a metadata that signals the power consumption per resolution of the current segment. This information guarantees an appropriate choice of the segment version with respect to the power budget. For this third strategy, the MPEG Green Metadata standard defines a metadata indicating the possible power savings per resolution for a segment representation. This proposal aims at controlling the induced QoE by lowering the resolution or the bit-rate. The transition between scenes is then controlled and the devices can act proactively [Duc15].

### Green Metadata Encoder Power Reduction

While the first Green Metadata proposals have focused on the decoder side, this last proposal provides means to reduce the encoder power consumption. Its goal is to use a quality recovering mechanism for low-quality encoded segments that are power minimized. A metric giving the quality of the last frame of each segment is sent as metadata to the decoder. This technique is called cross-segment decoding (XSD) [WLL+13].

An XSD decoder utilizes the quality metrics embedded in the high-quality segments to improve the decoding of the low-quality segments, producing a visual experience with a higher QoE that might not have been utilized by the encoder to increase the quality of the decoded video sequence for application.

The main drawback of this method is that it uses a closed-loop top architecture with a direct link between the decoder and the encoder. Therefore, a low power decoder is only possible with the associated encoder.

### 3.5.3 Conclusion on Low Power Decoders

At first, it shall be noted that standardization bodies such as MPEG put an effort in facilitating the emergence of energy efficient design. Indeed, it is a critical point for fast time to market of new technologies like HEVC.

It shall be also noted that software-based decoders, coming with much design flexibility, take up the challenge of future proof design. New features like HDR are bound to emerge in a near future and will integrate compression standards, making them evolve quickly. The decoders shall, somehow, take into account the evolutions and be flexible.

Current proposals encourage designers to follow this direction as proof-of-concept and confirm the feasibility of software-based solutions for HEVC [CAMJ14, LCW+15]. However, many of them are not optimized to tackle at the same time real-time management and energy efficiency, whereas modern SoC propose efficient tools to do so as described in Chapter 2. Complete system modeling from the platforms characteristics to the application needs is then required to design efficient systems.

This thesis proposes in Chapter 4 and Chapter 5 new approaches to use efficiently platforms and tools adapted to application needs.

Moreover, the MPEG standardization body also opens the door to new mechanisms to reduce the decoder power consumption using Green Metadata. However, the only way for such a decoder to reduce the decoding complexity is to change the input bitstream, leading to closed-loop systems. Conversely, Chapter 6 proposes a new decoder architecture capable of decoding unmodified input bitstreams with a reduced power budget.

# Part II

# Contributions

CHAPTER 4

# Energy Efficient Rapid Prototyping of Signal Processing

## 4.1 Introduction

### 4.1.1 Motivation

The very early stages of development are critical to ensure the success of a project. Among them, the architecture definition phase is crucial. In particular, it shall find the best match between the application itself and its implementation.

Various frameworks exist to assess these points. They can either focus on the application itself (e.g. Matlab® [Mat12]) or on the electronics with Methodology for the Conception of System of Electronics components (MCSE) [Cal96]. They seek different goals: improve the expressiveness and give a top-level view, put together system requirements with architecture level features, investigate the design space, etc. Once the first step of feasibility is attained, the priority shifts at system level to cost and efficiency. The designer shall then take into account the application QoS requirements and the platform specific characteristics. The challenge is to take into account all this information to propose a methodology and design efficient systems. The design output shall meet all the different requirements from the application QoS to the hardware features.

Among the most resource consuming applications, signal processing ones demand high computing resources with intensive calculations. It can also be coupled with real-time requirements. Therefore, to be feasible, these applications often rely on parallel executions. But at the same time, with the exponential growth of the data mobility, most of the devices that embed signal processing applications are handheld electronic devices. On top of just processing data, they shall also be energy efficient and use low power techniques. Especially, determining the best operating point between parallel execution and frequency scaling is not a trivial task.

The objective of this chapter is to provide a comprehensive framework that gathers the low power techniques provided by modern MPSoC with application characteristics. It also focuses into expressing the problem into a compact form that can be solved in polynomial time. Indeed, at early stages of the design phase, the design space shall be investigated rigorously within a short period of time. Therefore, we propose to constrain the problem formulation into an extended convex form to guarantee a reliable, rapid resolution and optimal solution.

### 4.1.2 Related Work

The high benefits of low power techniques, including DVFS and DPM, play an ever more significant role for embedded systems as they target low power and low energy consumption. Considering multicore architectures, the design space is even bigger and, as previously described in Section 2, parallel processing can be used to lower energy even more. For example, Piguet *et al.* [PMvKS95] explain that a perfect two-fold parallelization can be used to reduce the frequency (and voltage) by two and, therefore, lead to a reduction of one-quarter of the energy consumption.

Straightforward energy saving techniques exploit the available slack time with DVFS. They have been extensively studied as they ideally suit real task scheduling. Much of the previous work focus on monoprocessor architectures [IY98, KSY+02, SKL01] and consist in adapting the scheduling to *never-idle* while processing. To go further, Mishra *et al.* [MRZ+03] propose to analyze the degree of parallelism in a schedule. In this way, the non-active cores can go into a sleep modes. Bhatti *et al.* propose to build a hybrid power management called *HyPowMan* [BBA11] based on a machine-learning mechanism. The advantage of the proposed method is to learn online the characteristics of energy consumption DVFS and DPM mechanisms. All these proposals are heuristic-oriented energy-aware scheduling that propose to improve the energy consumption on-line by reducing the instant power consumption. In these approaches, the different elements composing the application are already known. The application degree of parallelism is not changed and these studies focus on the mapping on individual elements. This chapter proposes to shift the analysis at the early stage of the design phase to build an analytical model of the energy and explore the design space. The application is assumed to be parallelized by an adequate transformation. This model relies on dataflow modeling and, more particularly, on Single Rate Synchronous Data Flow (SDF) transformation [EL87].

Aupy *et al.* [ABDR13] propose several DVFS strategies for real-time systems under bounded execution time with proofs of optimality with convex modeling. However, it only considers the dynamic power part of the overall power consumption, which limits the scope of the proposed approach especially because modern MPSoCs are not only dominated by dynamic power but also by leakage power. By constructing an analytical model that derives the energy consumption, this chapter provides a more complete perspective than only power modeling. It is essential for battery-powered devices. It also targets convex modeling throughout the framework. This modeling makes it possible to solve the optimization problem with convex programming techniques.

To build a reliable energy model, Nelson *et al.* [NMM+11] propose to use dataflow Model of Computation (MoC) and find the best working processing frequency. Firstly, monocore implementations are targeted to build the model and express a frequency scaling strategy per actor composing the dataflow application. In a similar fashion, De Langen *et al.* [DLJ06] introduce the notion of energy modeling for DPM and DVFS systems to minimize the leakage power. They propose a *schedule and stretch* approach for MPSoC based on heuristics. However, and for both approaches, the parallelism level of each actor is not considered as an optimization parameter despite the dataflow model and focuses on mapping. The level of parallelism of each part of the scheduling is already set and the objective is to map in an energy-aware fashion. In this chapter, the parallelism level is an actual optimization parameter that is integrated into a complete formulation. Perfect and limited parallelisms are analyzed to cover a wide range of applications.

To minimize the total energy consumption in an MPSoC, the approach proposed by Chen *et al.* [CHK14] provides an optimal schedule and a frequency assignment for each actor. The energy minimization problem is solved thanks to Mixed Integer Linear Pro-

gramming (MILP) technique. A refinement technique has been developed to reduce the optimization time. This approach proposes an optimal DVFS and DPM combination, but like the other works presented above, the parallelism level of the application is not taken into account in the energy minimization formulation. This chapter proposes to determine the maximum energy improvement related to the parallelization of each actor and is not reduced to a one-actor approach.

By taking into account the speed-up due to parallelization, the approach proposed by Cho *et al.* [CM10] tackles the same optimization problem as the one considered in this chapter. A simplistic application model based on Amdahl's law is used to characterize the parallelism. The application is split into a serial section, executed by one core, and a parallel section executed by any number of core (fully parallelizable). In our work, a more complete formulation is proposed for multicore implementations accounting for the parallelism level of each actor. The proposed formulation maintains the convexity properties enabling convex solvers to find an optimal scheduling and energy management policy.

Furthermore, the previous work [CHK14, DLJ06, NMM+11] is based on the analytic model proposed by Jejurikar *et al.* [JPG04]. The power consumption is modeled as a function of different technology parameters. Most of the time these technology parameters are not available and can not be deduced easily from real measurements. Consequently, the usability of these models based on technological parameters is limited. *HyPowMan* [BBA11] proposed by Bhatti *et al.* uses on-line machine learning algorithm to optimize the scheduling strategy. However this technique cannot be used to perform design-space exploration at the early stage of the design. In this chapter, a power consumption model is computed from real measurements of the targeted platform with no pre-requisite information for the SoC maker. The power model is expressed as a function of the processing frequency and the number of active cores. It is back fitted with a generic approach that generates a convex formulation.

Finally, in this chapter, we address the problem of energy efficiency for MPSoC equipped with DVFS and DPM for applications under real-time constraints. In addition to the above work, we propose a model that integrates frequency scaling, deep sleep and parallelism at the same granularity level. These various elements have been incorporated while maintaining the convexity property of the model. Consequently, an optimal solution is obtained quickly thanks to the use of convex solver. The model is used in an optimization problem that can be solved with the  technique, an extension of the usual convex programming. In other words, we analyze how the designer can gather the application requirements and the platform characteristics to design systems focused on energy efficiency in a single framework. The problem is multi-dimensional and several parameters need to be considered: parallelism level, processing load, real-time requirements and low power techniques.

The rest of the this chapter is organized as follows. Section 4.2 presents how a MoC and, particularly, the SDF, is a powerful tool to analyze signal processing applications. Combined with knowledge on platform characteristics, it leads to the definition of rapid prototyping frameworks. Section 4.3 proposes to take into consideration the energy model for static dataflow applications. The generalization of this model to MPSoC is described in Section 4.4. Section 4.5 proposes a case study of HEVC offline decoding before discussing this chapter in Section 4.6.

## 4.2 Assisted Design of Signal Processing Applications

In this section, a context recall is given on how signal processing applications can be designed efficiently with appropriate tools. Especially, it looks for the parallelism level of the considered application that can exploited to improve the energy efficiency.

### 4.2.1 Models Of Computation

High level representations using block diagrams are often used to extract essential information of the considered application. To assist the designer, more formal models can be generated and are popular for the development of software and hardware systems. Using a specific semantics to describe the behavior of the application and the link between the different elements composing it can be a powerful way to design efficient systems. A MoC is a set of operational elements that can be put together to represent the application behavior. There exists a wide set of MoC, Figure 4.1 recalls some of the major ones [Ric97].

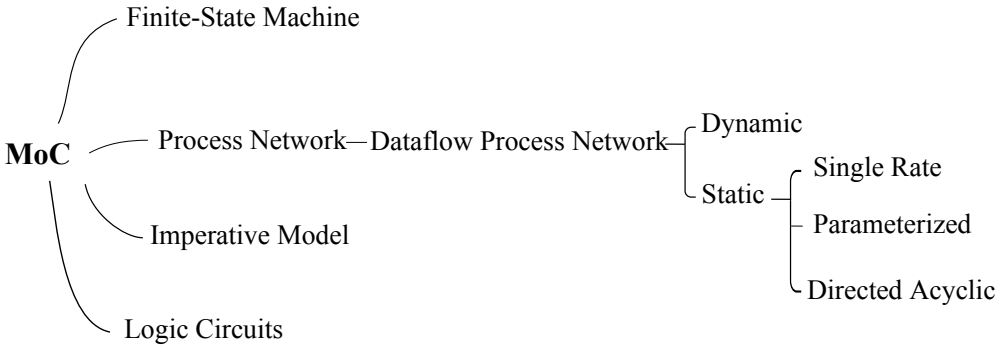


**Figure 4.1:** *Examples of Models of Computation*

- **Finite State Machine (FSM)** MoC semantics gathers a set of exclusive states that a system can hold.
- **Imperative** MoC proposes to execute sequentially modules to accomplish a task.
- **Logic Circuits** MoC relies on two elements: gates and boolean signals. The system has no state and the output of a gate only depends on the current value of the input value.
- **Process Network** MoC is a natural way used to describe signal processing systems where infinite streams of data are incrementally transformed by modules. It can run sequentially or in parallel.

In **Dataflow** MoC, modules are triggered or fired by the availability of data at inputs. Modules are called actor and can embed a hierarchical description [PBR09]. The schedule can be performed either dynamically or statically.

Considering static scheduling, more information can be extracted and used to enhance design. For example, Single Rate SDF can be used to express more parallelism of an application. Design from these MoC can benefit from their determinism, predictability or reconfigurability and consequently can be used to define a rapid prototyping framework. This framework explores the design parameters to find the best match between the underling platform and the targeted application.

From the application point of view, it needs to fulfill a set of requirements to fit into the MoC prerequisite. Figure 4.2 recalls how signal processing applications may be sorted. They can be either dynamic, i.e. the processing load and the time of execution cannot be predefined, or static. A static behavior brings to the designer properties to ease architecture design phase. It can be categorized at different levels and it ensures a known time of execution. The considered application can be a single algorithm or a sequence of applications. This latter use case is present widely for signal processing applications either in the digital communications area or in the image processing area.

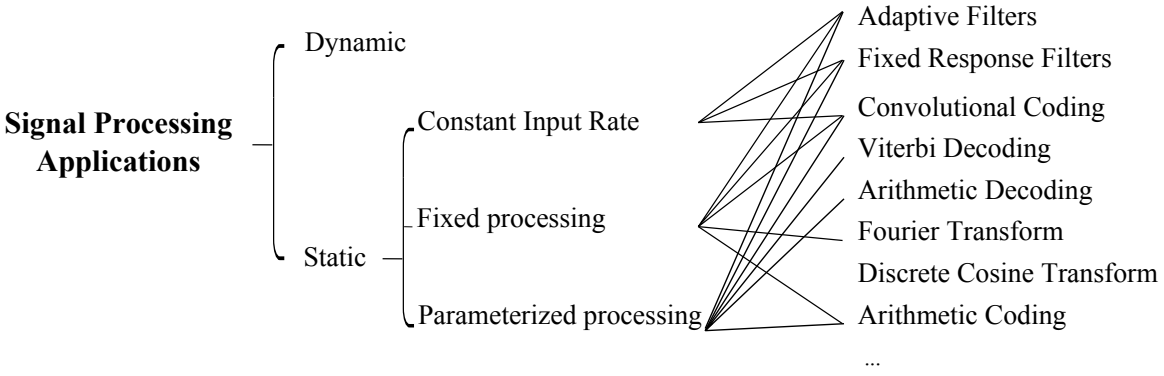


**Figure 4.2:** *Signal applications categories*

### 4.2.2 Rapid Prototyping

Cooling and Hughes [CH89] present the basic concepts of rapid prototyping in computer science. On the one hand it relies on the accurate description of the applications and its requirements. On the other hand it also needs the characteristics of the targeted platform. In the end, the process can be assisted by a dedicated tool and shall provide the best match between the application and its platform. Several criteria can be targeted for the optimization phase like throughput maximization, memory foot-print minimization or energy efficiency.

Desnos in [Des14] reviews the different rapid prototyping tools and gives a synthetic diagram of the different steps within the design process 4.3.

As shown in Figure 4.3, the designer inputs the system with the applications model, the requirements of the system (e.g. real time constraints) and the platform characteristics (e.g. processing capabilities). The independence between these three inputs of design phase eases the development of applications onto several architectures or the re-use of the architecture for several applications. The Algorithm-Architecture Matching (AAM) relies on these properties [GS03]. Therefore AAM is a promising methodology to design energy efficient systems as it can spot precisely the best setting point of the architecture.

Reducing the scope of applications, dataflow programming is widely used for signal and image processing. During the prototyping phase, the key properties can be extracted. Among them, the intrinsic parallelism can be exposed and used for efficient mapping on MPSoC [PPW+09]. At the end of the *mapping and scheduling* of Figure 4.3, the behavior of the different actors that compose the application can input a Gantt chart. This chart recalls the execution step [HPD+14]. Precisely it exhibits how parallel is the application with time.

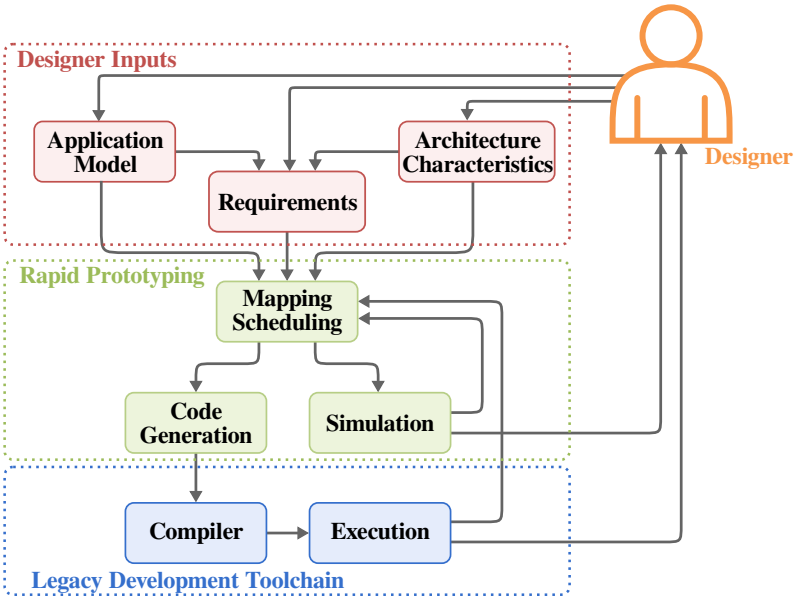


**Figure 4.3:** *Overview of a rapid prototyping tool [Des14]*

### 4.2.3 Assisted Design using Algorithm Dataflow Modeling and Online Power Optimization

For applications that demand high processing resources, only highly performing platforms can be envisioned. If the system is constrained by a deadline, it is bound to process the application at a sufficiently high frequency to fulfill this requirement. And because the processing frequency cannot be extended infinitely, architecture based on MPSoC and parallel processing can be the key for a fast time to market introduction. However, strict parallelism cannot be achieved for the complete execution and the actual pattern is a series of sequences with different levels of parallelism.

In this section, we propose to couple online optimizations of resource allocation (processing frequency and core allocation) with the parallelism level estimation obtained at compile time. Thanks to an accurate estimation of the parallelism level obtained at compile time, the online optimizer can improve the resource allocation and reduce power consumption for a given QoS.

#### Framework Description for Power Optimization

In collaboration with S. Holmbacka, we propose in [HNP+14, HNP+15] a combined framework based on offline feature extraction and online optimization as depicted in Figure 4.4.

At first, a dataflow representation is used for its expressiveness to extract characteristics of the considered application. Indeed it increases the predictability of the application behavior. Here, it enables an accurate description of the parallelism over time. The PREESM tool [PDH+14] is used to extract parallelism level, also called *P-value*. This information can change with time and is made available to upper layers. Figure 4.5 depicts three different cases.

In extreme cases, the application can be either purely sequential or entirely parallel. In a more general case, the application is sometimes sequential, sometimes fully parallel

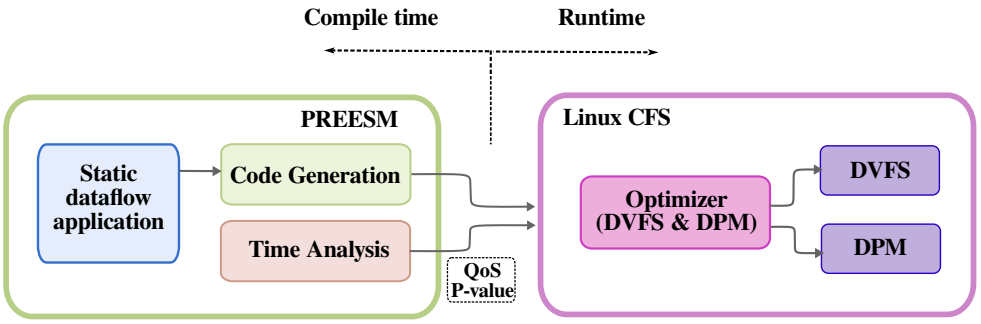


**Figure 4.4:** *Optimization framework proposed in [HNP+14]*

and sometimes partially parallel. It corresponds to the case *mixed-parallel application* of Figure 4.5.

*P-value* is in the range $[0.0, 1.0]$, where 0.0 is a serial sequence and 1.0 is a ideal parallel sequence for the used hardware platform. From Amdahl's law and the Gantt chart in Figure 4.5, the *P-value* of a semi-parallel application is computed as:

$$P\text{-}value = \left( \frac{\frac{1}{S} - 1}{\frac{1}{N} - 1} \right) \tag{4.1}$$

where $S$ is the speed-up factor between the sequential and optimized applications after parallelization and $N > 1$ is the total number of cores. Consequently a value of 0.67 describes a scalability to half of the processing elements. The *P-value* can furthermore be calculated as an average of the whole sequence or dynamically for each sub-sequence for enhanced precision.

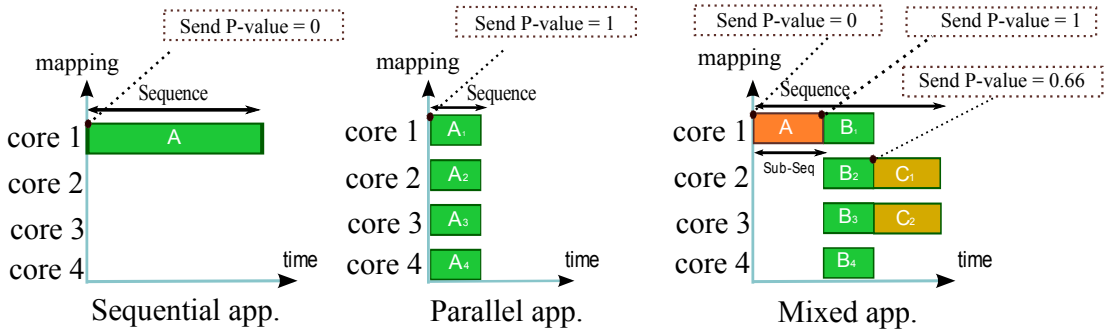


**Figure 4.5:** *P-value extraction in a sequential, parallel and mixed-parallel application*

In a second step, the *P-value* is used online in a runtime energy optimizer. The optimizer is proposed by Holmbacka *et al.* [HALL13] and finds a good tradeoff between the reduction of processing frequency and deep sleep management.

With resource-to-power model, the optimizer requires a model of application speed-up to minimize the energy. Two actuators, clock frequency scaling using DVFS and core activation using DPM, are described according to two separate models.

For simplicity, we model the performance of frequency scaling of an application as a linear combination of clock frequency $f$ as:

$$P_{\text{perf}}(f) = K_f \cdot f \tag{4.2}$$

where $K_f$ is a constant, which means that e.g. 2x increase in clock frequency models a double in speed-up.

In contrast to the simple approximation of performance to clock frequency, modeling the performance as a function of the number of cores is more difficult since the result depends highly on the inherited parallelism and scalability in the application. Because of this issue, the *P-value* is introduced. The performance $P_{\text{perf}}$ model for the number of active cores is modeled according to Amdahl's law as:

$$P_{\text{perf}}(c) = K_c \cdot \frac{1}{(1-P) + \frac{P}{c}} \tag{4.3}$$

where $K_c$ is a constant, $P$ is the parallelism expressed as the *P-value* and $c$ is the number of available cores. The parameters $K_f$ and $K_c$ are selected to normalize the performance parameters to avoid rounding errors in the optimization algorithm. Amdahl's law models a high performance increase with number of cores as long as the number of active cores is low but decreases as the number increases. Hence, the speed-up becomes sub-linear as more cores are added, and eventually increasing clock frequency becomes more energy efficient. The total performance is the sum of the two models as:

$$P_{\text{perf}}(f, c) = P_{\text{perf}}(f) + P_{\text{perf}}(c) \tag{4.4}$$

### Setup of the NLP Optimization Solver for Power Optimization

With a complete mathematical representation of the system, both the power and performance model are integrated into a NLP (Nonlinear Programming) solver and used to determine the minimum power dissipation under performance criteria. The method implements the Sequential Quadratic Programming (SQP) [GMMS97] solver with the plain objective function and side constraints given in Equation (4.5).

The objective of the solver is to determine an actuator configuration which minimizes the power $P_{tot}$ while still providing sufficient resources to all applications. The required performance is given by the programmer as a *setpoint* $S$, and the actual performance $R$ is monitored and transmitted to the power manager. The lack of resources in the applications is calculated as the difference between actual performance and the setpoint and appears as a positive error value $\varepsilon$ in the optimizer i.e. $\varepsilon = S - R$. For example, a video player using a setpoint of 25 fps (frames per second) but actually measuring 20 fps produces an error of $\varepsilon = 25 - 20 = 5$. The application can request more resources by setting a lower bound of QoS limit $Q$, which indicates the lowest tolerable performance e.g 24.5 fps.

The power optimization problem is defined as follows for each application:

$$\begin{aligned} \underset{f,c}{\text{minimize}} \quad & P_{tot}(f, c) \\ \text{subject to} \quad & \varepsilon - (f + c) < S - Q \end{aligned} \tag{4.5}$$

where the variables: $f$ and $c$ are the actuators (DVFS and DPM). $S$ is the setpoint, $\varepsilon$ is the error value and $Q$[1] is the lower QoS limit. The optimization rule states to *minimize power while eliminating enough errors such that at least the lower bound QoS limit is reached.*

- The setpoint $S$ is set by the user to represent a practical performance aspect of the application which should be reached

[1] $\varepsilon$ and $Q$ are normalized to the range in which $f$ and $c$ operate

- $\varepsilon$ is measured by the application – this is the current (real) performance.
- The QoS limit $Q$ can be set by the user or obtain a default value for example 95%. This means that a 5% deviation from the setpoint would be treated as acceptable.

### Results of the Power Optimization

The power manager including the NLP optimizer was evaluated on the Exynos 5410 mobile multicore platform as described latter in Section 4.3.2. The power measurements are performed with embedded power sensors.

A Sobel filter is used for performance evaluation since it presents data parallelism and is a good application to explore architecture strategies that can be parallelized for the filtering part and sequential for any preprocessing function [KAN+11]. Using our design framework, we add performance requirements on the filtering to match the intended playback rate of 25 frames per second (fps) with an additional small safety margin, i.e. 26 fps, to ensure that no frame miss would occur during the playback. This means that the application filters frames with a rate of 26 fps and sleeps for the remaining (very short) time window.

The three applications were generated automatically using PREESM tool:

1. **Fully sequential.** A single-threaded sobel filter was generated and $P = 0.0$ was injected for the complete execution since the application cannot be split onto multiple cores.
2. **Fully parallel.** A multi-threaded sobel filter was generated. The picture on which the filter is applied can be divided into several slices without internal data dependencies [KAN+11] which generates a close to perfect parallel execution. $P = 1.0$ was hence injected for the complete execution.
3. **Mixed-parallel.** A sequential preprocessing filter was applied in sequence with the parallel sobel filter. With a phase dependent parallelism two scenarios were considered:
   (a) **Static P-value.** An average *P-value* according to Equation (4.3) was injected for the complete execution.
   (b) **Dynamic P-value.** A per-phase defined *P-value* was injected as $P = 0.0$ for the sequential phase and $P = 1.0$ for the parallel phase.

Table 4.1 shows the total energy consumption for all use-cases and the energy savings by using the online optimizer in the last rows. The energy reduction is the result of allowing applications to better express intentions and behavior.

By fine tuning the application to use dynamic *P-values*, the energy consumption can be further decreased as the optimizer is able to scale the hardware closer to software requirements.

| | Serial | Parallel | Mixed | |
|---|---|---|---|---|
| | | | static *P-value* | dynamic *P-value* |
| Energy - Linux OnDemand | 281.24 | 261.44 | 527.73 | 527.73 |
| Energy - proposed solution | 175.33 | 117.36 | 386.70 | 275.71 |
| Energy savings | 37.7% | 55.1% | 26.7% | 47.8% |

**Table 4.1:** *Energy consumption (Joules) for a 5 min run of 3 different applications*

#### Discussion

Contrary to traditional methods that promote parallel software and *As-Fast-As-possible* execution as energy efficient, the proposed framework shows the limits of energy efficient parallelization on MPSoC. The workload applied on such a CPU should be only parallelized to the point at which the static power caused by enabling cores starts to dominate over the dynamic power caused by frequency scaling.

It shall be noted that the proposed method needs a perfect knowledge of the application capabilities and characteristics. Thus, these characteristics shall not change in time, hence are considered as static. On top of that, an NLP solver is used. Issues with nonlinear problems are firstly the inability to ensure global optimum, and secondly, a high complexity with respect to the control variables.

In the rest of this chapter, the modeling abstraction is shifted to a higher level where the application is considered as a black box. The purpose is to propose an energy model that can be used to formulate the optimization problem for obtaining an energy efficient design. Because energy is a convex function of the processing frequency, this latter problem shall be solved with convex programming technique. This energy optimization process is done at design-time. It is proposed then to start with the basics of the energy model and application assumptions. The model is then enriched with low power design techniques like DVFS and DPM and extended to MPSoC. This model is then incorporated into a rapid prototyping framework that defines the best operating point from an energy point of view. The proposed operating point is compared to *remarkable* points of the design space. Finally, we show how it can be used to propose an energy efficient HEVC decoder for offline decoding.

## 4.3 Energy Optimization of an Application Described with a Dataflow Model of Computation

Contrary to the framework presented in previous sections, the approach developed in this section addresses the energy optimization of a system at design time. The purpose of this work is to define, at an early stage of the design phase, the most energy efficient operating point using the AAM methodology. The application is considered as a *Black Box*. The term *Black Box* signifies that the exact behavior of the application is not needed. The only required information for the optimisation is the overall load expressed in cycle counts. A more general problem is studied where a pipeline of *Black Boxes* is executed with a requirement of global deadline.

Firstly, energy modeling is introduced in a general context and then used to handle real time applications on a single core. Secondly the problem is extended to MPSoC and confronted to the characteristics of a real platform.

### 4.3.1 Description of the Energy Model of an Application Considered as a "Black Box"

#### Modeling the Processing Energy

The power consumption of SoCs has been thoroughly studied in the literature [RCN02, Pig04] and several models have been proposed with good accuracy. For this work, we use the power model from [JPG04] where the total power consumption is given by Equation (4.6) and detailed in Equation (2.4). At the system level, several solutions exist to tackle the static and dynamic sources of energy consumption. We propose here to use a

rapid prototyping phase to find the best match between what the platform offers and what the application demands.

$$P_{tot} = P_{dyn} + P_{stat} \tag{4.6}$$

To simplify the model and make the study more general, the following conventions are used. The key parameters such as processing frequency $f$, the number of cores $c$, the energy and power are used either with their actual value or with a normalized notation. As an example, Equation (4.7) expresses the normalized frequency $f$ from the actual frequency $f_{proc}$ with $f_{proc} = [f_{min}; f_{max}]$. $f_{max}$ refers to the maximum operating frequency of the system in a context of DVFS.

$$f = \frac{f_{proc}}{f_{max}} \tag{4.7}$$

### Energy per Cycle using DVFS and DPM in a Monocore Execution

Jejurikar *et al.* [JPG04] proposed an analytic model of the power consumption as a function of technological parameters. This model has been used in several energy efficiency studies [DLJ06, NMM+11].

Energy modeling can be illustrated with a simplified version of this model: $P_{dyn}(f) = C_{eff}.f^2$ and $P_{stat} = Constant$ from Equation (4.6). The left part of Figure 4.6 shows the power dissipation as a function of the processing frequency. The total power dissipation is dominated by the dynamic power and is a strictly increasing function of frequency. As a consequence, the most *Power-efficient design* corresponds to processing at the minimum frequency $f_{min}$ respecting the execution requirements.

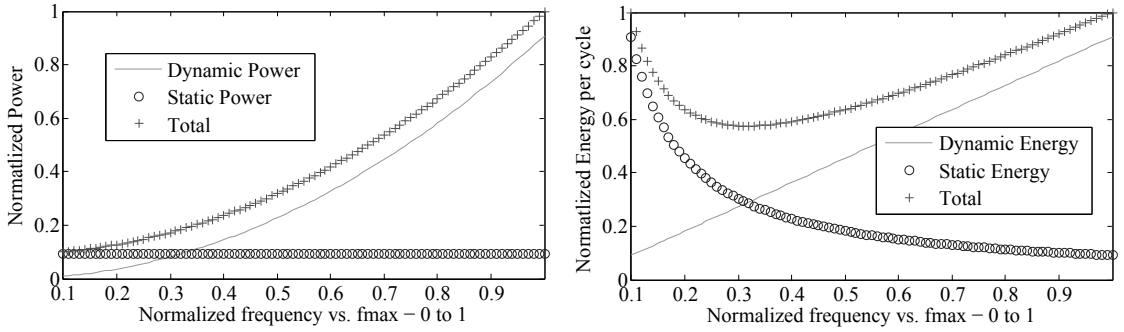


**Figure 4.6:** *Left: Normalized Power as function of the normalized processing frequency - Right: Normalized Energy as function of the normalized processing frequency.*

The capability of the system to shut down processing cores is an essential feature to design energy efficient systems. One can argue that minimizing the power consumption of a process leads to the most energy efficient set-up. However, this is not entirely true for systems where core shut down mechanisms (or DPM) are implemented. For this latter case, the energy model can be defined as the energy needed per cycle count of processing. It is determined in Equation (4.8) as a function of the processing frequency.

$$E_{cycle}(f_{proc}) = \frac{1}{f_{proc}T}. \int_0^T P_{tot}(t)\,\mathrm{d}t \tag{4.8}$$

with $P_{tot}$ taken from Equation (4.6) and $f_{proc}$ the processing frequency.

Using equation 4.8, a model of processing energy can be extracted. Contrary to the power model of Figure 4.6 that shows that the minimum power dissipation happens when using the minimum frequency, the energy model exhibits a critical frequency $f_{efficient} = 0.3$ that provides a better energy efficiency than $f = 0.1$. Therefore, using this frequency as the processing frequency is optimal in terms of energy, providing that the system can shut down cores once they become idle.

Considering a real-time application which iterations shall be terminated before a deadline, three types of scheduling can elaborated from an energetic point of view :

- the *As-Slow-As-Possible* scheduling runs at the slowest possible frequency to respect the deadline. This scheduling minimizes the dynamic power consumption and relies on DVFS techniques.
- the *As-Fast-As-Possible* scheduling runs at the maximum possible frequency and then rests (shutting down idle cores). This scheduling minimizes static power and relies on DPM technique.
- *energy-efficient* scheduling that runs at the $f_{efficient}$ from Figure 4.6. This technique relies on both DPM and DVFS techniques.

These three policies are summarized in Figure 4.7.

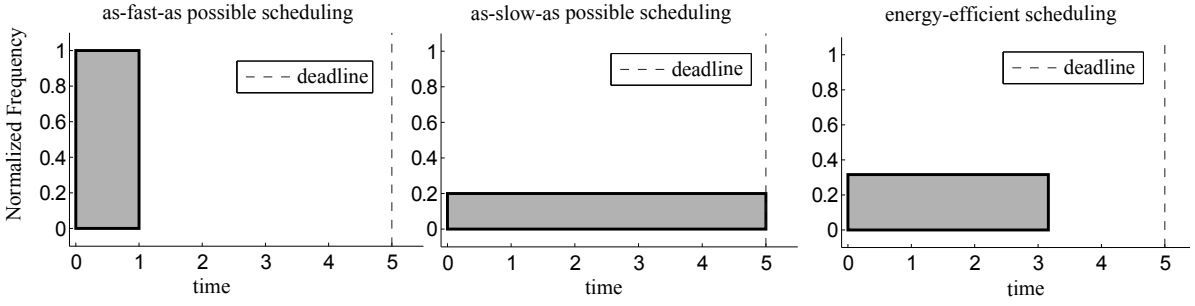


**Figure 4.7:** *Comparison of scheduling strategies - left: As-Slow-As-Possible scheduling - center: As-Slow-As-Possible - right: Energy Efficient scheduling*

### Case of a Real-time Application with a Static Behavior

From a formal point of view, the minimization of the energy of a real-time system can be formulated as an optimization problem. It consists in minimizing the overall energy consumption $E_{tot}$ while ensuring the application output delivery before the deadline. This minimization problem is formalized in Equation (4.9). From a rapid prototyping point of view, the processing frequency $f_{proc}$ is the parameter to be tuned, assuming that the processing load is known.

$$\begin{aligned} \underset{f_{proc}}{\text{minimize}} \quad & E_{tot}(f_{proc}) \\ \text{subject to} \quad & \frac{w_{proc}}{f_{proc}} \leq D \\ & f_{proc} \geq f_{min} \\ & f_{proc} \leq f_{max} \end{aligned} \tag{4.9}$$

with $f_{proc}$ the processing frequency in Hz, $w_{proc}$ the load in cycles of the application, $D$ the maximum time allowed to execute the application in seconds, $f_{min}$ the minimum processing frequency in Hz supported by the platform and $f_{max}$ the maximum processing frequency in Hz supported by the platform.

Let us illustrate the scope of the problem with a video processing application. The application can be represented at top level by the sequence displayed in Figure 4.8. The first actor on the left extracts information relative to a frame from a bitstream, the decoding process decompresses information and the rendering process provides the decoded frame to a display device. The graph is executed for every frame decoding and display. The deadline arises from the playback rate and is expressed in frame per second (fps). The complexity (amount of work) of the processing can be expressed globally by $w_{tot}$ or per actor (function) by $w_i$. Let $\mathbf{w} = [w_0, \ldots, w_{N-1}]$, be a N-length vector defining the complexity of the $N$ actors of the application.

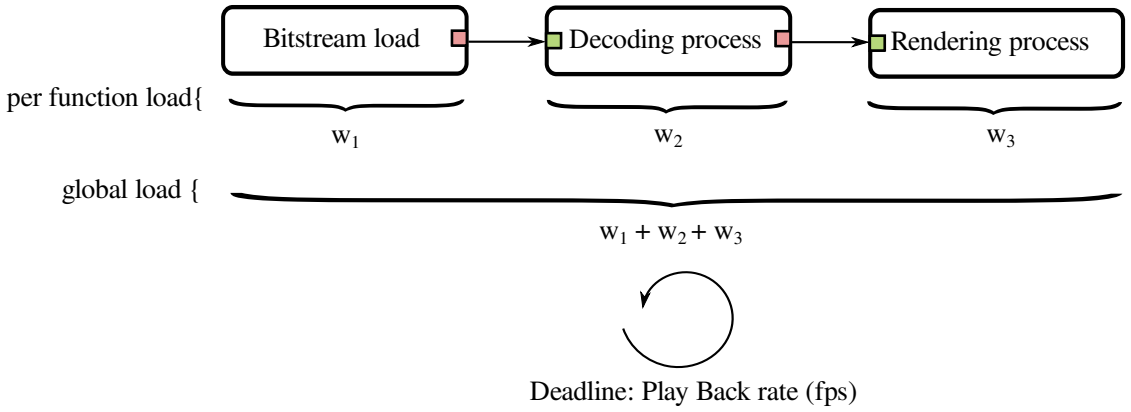


**Figure 4.8:** *Generic video decoding graph*

The optimization of Equation (4.9) is reformulated in Equation (4.10) by integrating the energy-per-cycle model. This new formulation accounts for the number of cycles $w_i$ for each actor $i$ when the system is capable of handling frequency changes over time. Having DVFS capabilities, the system can change its frequency before starting the execution of a new actor.

$$\begin{aligned} \underset{f_i}{\text{minimize}} \quad & \sum_{i=1}^{N} w_i E_{cycle}(f_i) \\ \text{subject to} \quad & \sum_{i=1}^{N} \frac{w_i}{f_i} \leq D \\ & f_i \geq f_{min} \\ & f_i \leq f_{max} \end{aligned} \tag{4.10}$$

with $N$ the number of actors composing the application, $f_i$ the processing frequency of actor $i$ in Hz, $D$ the deadline for application completion in seconds. $f_{min}$ is the minimum processing frequency, $f_{max}$ is the maximum processing frequency. Let $\mathbf{f} = [f_0, \ldots, f_{N-1}] \in \mathbb{R}^N$, be a N-length vector defining the frequency associated to the $N$ actors of the application.

### Solving the Energy Optimization Problem with a Convex Programming Technique

The proposed energy per cycle function has the property of being convex, since the set of its possible outputs fulfills the convex property. A subset $\mathbf{C}$ of $\mathbf{R}^n$ is said to be **convex** if and only if it contains **the line segment between any two points chosen in C**, as shown in Equation (4.11).

$$\forall y_1, y_2 \in \mathbf{C}, \forall \lambda \in [0,1] \ : \ \lambda y_1 + (1-\lambda) y_2 \in \mathbf{C} \tag{4.11}$$

Figure 4.9 recalls the different optimization technique classes with their related assumptions. The energy efficiency optimization problem falls into the category of convex problems.

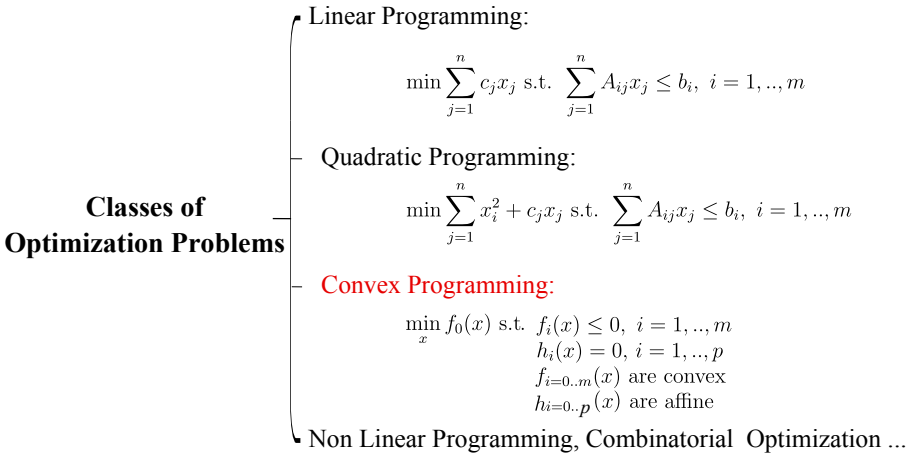


**Figure 4.9:** *Classes of optimization techniques*

Therefore, the *Disciplined Convex Programming* technique [BV04] is used here to solve the problem stated in Equation (4.10). Convex problems are proven to be solved in polynomial time. The output values are obtained with the `CVX` tool [GBY08], a convex optimization solver. The energy model and constraints are given as formulated in problem (4.10).

To illustrate how the optimization problem can be solved, we consider the following case. A basic application is depicted in Figure 4.8. The platform model is taken from the Figure 4.6 with the normalized energy model of Equation (4.8). The objective is to find the processing frequency vector $\mathbf{f}_{opt}$ that minimizes the energy needed per execution of one iteration subject to three constraints: the iteration shall end before the pre-defined deadline $D$ and the processing frequency shall be greater (resp. smaller) than the minimum processing frequency (resp. maximum processing frequency). The number of operations needed to execute a single iteration is set to 1. As depicted in Figure 4.7, this scheduling option is called *Energy Efficient* scheduling.

From an performance point of view, this scheduling is compared with the two other popular options of Figure 4.7. *As-Fast-As-Possible* scheduling runs the application at its maximum processing frequency. *As-Slow-As-Possible* scheduling runs the application at its minimum processing frequency that fulfills real-time requirements.

```
loop over deadline D{
    // Compute the processing frequency of Energy Efficient Scheduling
    f_eff = convex_optimization(D)
    Energy_convex = compute_energy(energy_model,f_eff)
    // Compute the processing frequency of As-Fast-As-Possible
        scheduling
    f_afap = fmax
    Energy_afap = compute_energy(energy_model,f_afap)
    // Compute the processing frequency of As-Slow-As-Possible
        scheduling
    f_asap = as_slow_as_possible(D)
    Energy_asap = compute_energy(energy_model,f_asap)
}
```

**Listing 4.1:** *Energy Simulations*

Figure 4.10 plots a performance comparison of the different strategies from the energy perspective as a function of the deadline. Deadlines are explored from a tight deadline, which is the time to execute the application at the maximum frequency, to a loose deadline, which is the time to execute the application at the minimum frequency. In all cases, the deadline is respected.

When the deadline is loosened from the tight deadline, the processing frequency of the *As-Slow-As-Possible* scheduling and *Energy Efficient* scheduling is lowered. The energy consumption is then improved. The energy reaches then a *most energy efficient* point from which there is no interest in reducing the processing frequency anymore. This points matches with the $f_{eff} = 0.3$ of Figure 4.6. From this point, the *As-Slow-As-Possible* scheduling continues to reduce the processing frequency when the deadline is loosened but it consumes more energy than the *most energy efficient* scheduling.

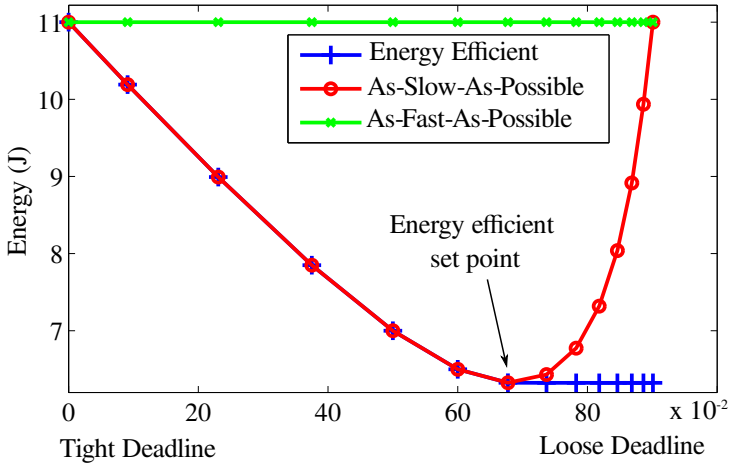


**Figure 4.10:** *Performance comparison of the different scheduling strategies as a function of the deadline position*

As a conclusion to this section, *Disciplined Convex Programming* is a powerful tool to design energy-efficient systems. On a *Black Box* level application that is executed on a platform with DVFS and DPM, we show that the energy optimal frequency is neither the *As-Slow-As-Possible* nor the *As-Fast-As-Possible* frequency set-up. The energy optimal frequency can be found with *Disciplined Convex Programming*, provided that an energy model of the underlying platform *with convexity* properties exists.

One may recall that the conclusions of this section rely on the theoretical energy-per-cycle model presented in Figure 4.6. The next section studies the problem with an energy-per-cycle model built from measurements of a real platform.

### 4.3.2 Framework Description for Minimizing Energy on a Real Platform

#### Energy Optimization on a Fully loaded Platform

Even though the optimization problem can be modeled simply, using the model to address a real platform is a challenging issue. Indeed, deriving the energy is generally presented as a model extraction from the SoC information [JPG04]. But in many realistic cases, precise information of the SoC power consumption is not available and thus not usable for the system designer in a rapid prototyping context. Using an inaccurate model is likely to lead to an inefficient design and to invalidate the method.

However, instead of using artificial power models, methods for direct power measurements are proposed in previous studies [RJS+13, BSB+14, HNP+14]. In next sections, the objective is to compute an analytical formulation of the power consumption from measurements.

##### *Building a Power Model from Measurements*

To highlight the methodology, an Exynos 5410 SoC is considered as a whole and the power consumption is measured for an *intensive* processing benchmark.

The power model is derived from power measurements using the `stress` benchmark [Wat]. This benchmark is widely used for energy characterization because of its capability to tune the stress parameters [HNP+14]. The benchmark is run under Linux on four threads on the four cores with all the available DVFS configurations provided by the Exynos SoC. In Figure 4.11, the power consumption is depicted as a function of the processing frequency and is normalized versus the maximum frequency $f_{max} = 1600MHz$. As expected, and closely to the power model from Figure 4.6, the minimum power state is attained when the frequency is minimal.

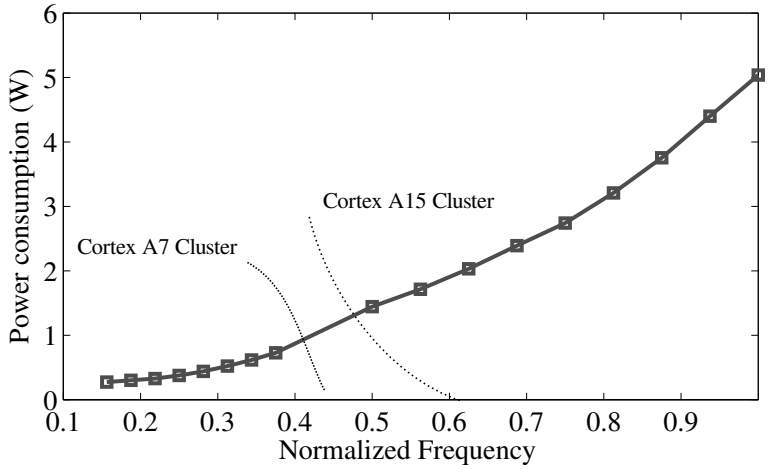


**Figure 4.11:** *Power consumption vs. operating frequency with 4 fully loaded cores on an Exynos 5410*

Section 4.3.1 establishes the relation between the power and the energy per cycle of processing. At the system level, the energy per cycle can be computed from power measurements as a function of the processing frequency.

Figure 4.12 depicts the energy behavior of the system. Contrary to the power, the most energy efficient state is not reached at the minimum frequency $f_{min}$. The optimal frequency $f_{eff}$ is 15 % more energy efficient than the minimal frequency for the Cortex-A7 cluster. It is also the case for the complete system that includes both a Cortex-A7 cluster and a Cortex-A15 cluster. These measurements confirm on a real SoC the initial model depicted in Figure 4.6.

In Figures 4.11 and 4.12, the full processing capacity of the architecture at each frequency is demanded. As a consequence, these figures do not reflect the capacity of the system to shut down individual cores but only its capacity for DVFS.

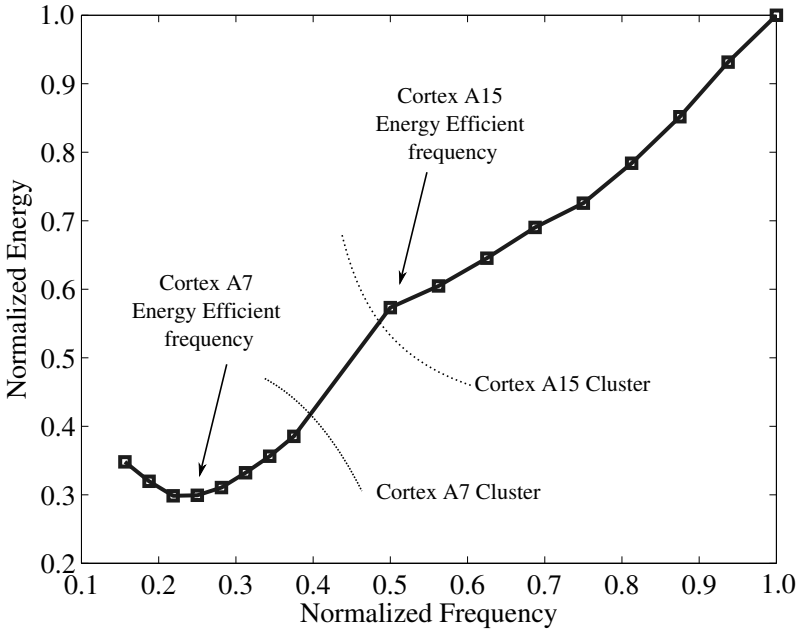


**Figure 4.12:** *Energy per cycle a function of normalized operating frequency of a fully loaded Exynos 5410*

The next step of building an energy consumption model for the system is to derive a mathematical expression from the power measurements and to use it as input of the optimization solver.

### Framework to Solve the Energy Optimization Problem for Real-time Applications

In Section 4.3.1, the *Disciplined Convex Programming* technique has been proposed as a solver for the real time system. The energy model provides the function to minimize that must be convex. However, in the context of a real platform, only measurements are available and a mathematical convex expression of the platform behavior is needed.

There are many fitting methods that can be used for model building. Ren *et al.* use linear regressions and Multivariate Adaptive Regression Splines (MARS) curve fitting in their energy models to associate complexity of processing to a mathematical model [RWJ+13, RJS+14]. Benmoussa *et al.* propose to use regression models to associate video parameters to complexity estimations [BBS+15a]. The objective of these approaches is to find the best match between the input parameters and the observed data. In this section,

there is an additional constraint to the model building. The functions linearly composed to feed the regression model should be convex to ensure the convexity of the overall model.

Given the general trend that is observed on power and energy characteristics, linear regression is an effective tool to model the system from observation points. It is commonly used, together with polynomial representations. For a given problem, the linear regression model of a set of measured data $y_i|i = (1, 2, 3, ..., n)$ from $x_{i1}, x_{i2}, x_{i3}, ..., x_{iq}$ input data is expressed by Equation (4.12).

$$\begin{aligned} y_1 &= a_0 + a_1x_{11} + a_2x_{12} + ... + a_qx_{1q} + \varepsilon_1 \\ y_2 &= a_0 + a_2x_{21} + a_2x_{22} + ... + a_qx_{2q} + \varepsilon_2 \\ &.. \\ y_n &= a_0 + a_2x_{n2} + a_2x_{n2} + ... + a_qx_{nq} + \varepsilon_q \end{aligned} \tag{4.12}$$

Using a matrix form, Equation (4.12) becomes Equation (4.13).

$$\mathbf{y} = \mathbf{Xa} + \boldsymbol{\varepsilon} \tag{4.13}$$

where $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ .. \\ y_n \end{pmatrix}$, $\mathbf{X} = \begin{pmatrix} 1 \; .. \; x_{1q} \\ 1 \; .. \; x_{2q} \\ .. \\ 1 \; .. \; x_{nq} \end{pmatrix}$, $\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ .. \\ a_q \end{pmatrix}$, $\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ .. \\ \varepsilon_q \end{pmatrix}$

With the Minimum Mean Square Error (MMSE) method, the Equation (4.13) can be reformulated to formulate the regression vector $\widehat{\boldsymbol{a}}$ as in Equation (4.14):

$$\widehat{\boldsymbol{a}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} \tag{4.14}$$

Therefore, an estimation $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{a}}$ of $\boldsymbol{y}$ is obtained. It shall be noted that in case of polynomial curve fitting, the $\boldsymbol{X}$ matrix uses polynomial expressions of parameter $x$ with :

$$\boldsymbol{X} = \begin{pmatrix} 1 \; x_1 \; .. \; x_1^q \\ 1 \; x_2 \; .. \; x_2^q \\ .. \\ 1 \; x_n \; .. \; x_n^q \end{pmatrix}$$

### *Generating the Convex Energy Model*

Convex functions play an important role in optimization problems as the output of the functions has no more than one minimum. They have particular properties among which [Roc70]:

- nonnegative multiple: $\alpha f$ is convex if $f$ is convex, $\alpha \geq 0$
- sum: $f_1 + f_2$ is convex if $f_1$ and $f_2$ are convex

The list below gathers some well-known convex functions on $\mathbb{R}$ :

- affine: $ax + b$ on $\mathbb{R}$, for any $a, b \in \mathbb{R}$
- exponential: $e^{ax}$, for any $a, b \in \mathbb{R}$
- powers: $x^\alpha$ on $\mathbb{R}_+$, for $\alpha \geq 1$ or $\alpha \leq 1$

The **framework** that generates the energy optimization problem formulation is presented in Figure 4.13. It consists of the following steps. Firstly, the **Platform measurements** step consists in extracting the platform characteristics as described in Section 4.3.2. Then, the **List of convex functions** step consists in choosing functions from the alphabet of convex functions shown above. As the affine composition of convex functions remains a convex function, the combination of several single functions is used to improve the fitting quality of the result.

During the model building phase, the **MMSE** step consists in finding the regression model of the platform from the actual measurements. Then the **Check Convexity** step verifies that the resulting function is still convex. Indeed in some cases the MMSE step can output negative parameters. In this case, it results in a violation of the convexity property and the designer shall change the list of convex functions. **Check quality** consists in verifying the performance of the fitting curve. The coefficient of determination $r^2$ is used as a figure of merit and is described in next section. The acceptability is set by the designer based on her/his requirements.

During the optimization phase, the **Formulation** step inputs the Disciplined Convex Programming tool `CVX` [GBY08] with the fitting function. Finally, the **Simulation** step runs the real-time scenario within `CVX` and outputs the energy simulation results to the designer.

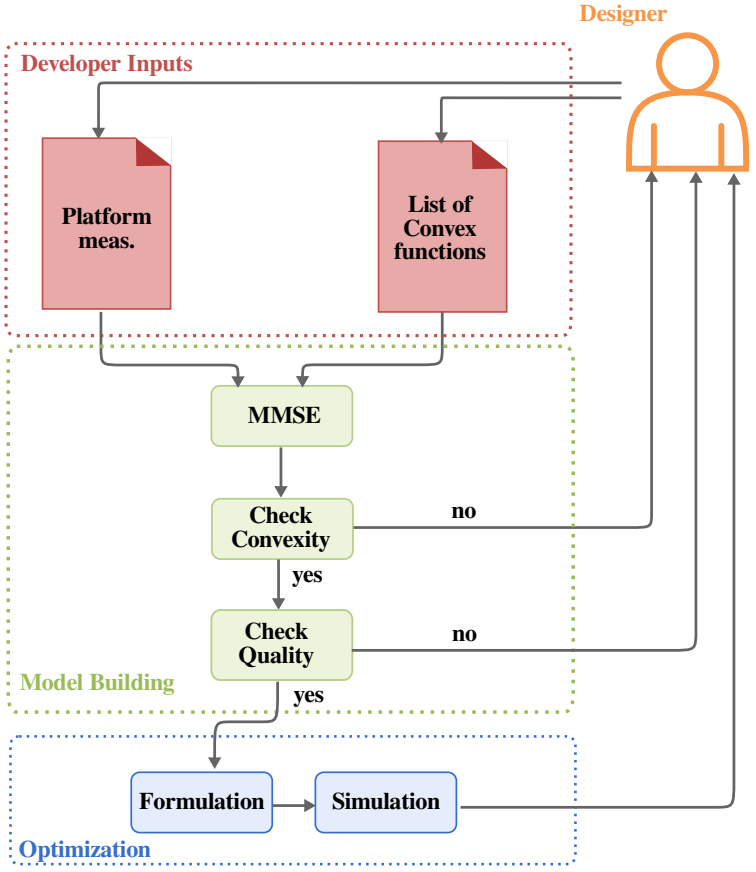


**Figure 4.13:** *Process to formulate the energy optimization model*

***The Coefficient of Determination $r^2$: A Reliability Indicator of the Fitting Curves***

Intuitively, the coefficient of determination $r^2$ represents the percentage of data that is correctly fit by the model. $1 - r^2$ is left unexplained and is equivalent to the $\varepsilon$ term of Equation (4.13). Therefore, $r^2$ represents how well the regression curve represents the measurements data and it is such that $0 < r^2 < 1$. $r^2$ is used to compare the different formulations of the $\boldsymbol{X}$ matrix based on measurements. Its expression is given by Equation (4.15):

$$r^2 \equiv 1 - \frac{\|\boldsymbol{y} - \widehat{\boldsymbol{y}}\|^2}{\|\boldsymbol{y} - \bar{\boldsymbol{y}}\|^2} = \frac{\|\widehat{\boldsymbol{y}} - \bar{\boldsymbol{y}}\|^2}{\|\boldsymbol{y} - \bar{\boldsymbol{y}}\|^2} = \frac{\mathrm{var}(\widehat{\boldsymbol{y}})}{\mathrm{var}(\boldsymbol{y})} \tag{4.15}$$

where $\bar{\boldsymbol{y}}$ is the average of the measured data $\boldsymbol{y}$ and $\widehat{\boldsymbol{y}}$ is the value generated by the model [DSP66, GS90].

**Step-by-step Execution of the Framework for Characterizing an Octa-core Exynos 5410 SoC**

In this section, the energy modeling framework is applied to a real platform that is intensively used in this document: an octa-core Exynos 5410 SoC. This SoC embeds a big.LITTLE configuration with four ARM Cortex-A15 cores and four ARM Cortex-A7 cores providing a set of 17 DVFS configurations from 250 MHz to 1.6 GHz. On this platform, only four cores can run simultaneously. From 250 MHz to 600 MHz, the SoC uses the Cortex-A7 cluster. From 800 MHz to 1.6 GHz, the Cortex-A15 cluster is used.

The main characteristics of the platform, in terms of hardware and software, are shown in Table 4.2.

| | |
|---|---|
| SoC | Samsung Exynos5 Octa 5410<br>ARM Cortex-A15 Quad 1.6 GHz max.<br>ARM Cortex-A7 Quad 600 MHz max. |
| Memory | 2GB LPDDR3 @ 800MHz |
| OS | Ubuntu 14.04<br>with GCC 4.8.2 |
| Energy sensors | INA 231<br>Measurement frequency 10 Hz |

**Table 4.2:** *Experimental configuration*

The power measurements were previously displayed in Figure 4.11 for a fully utilized system. To illustrate the flow presented in Figure 4.13, Table 4.3 shows the output of the MMSE step from four different well performing examples of fitting functions.

The fitting functions are the results of the system expressed by Equation (4.14). For clarification purposes, the model 1 is taken as an example. The different steps that are necessary to output the fitting model 1 are given hereafter. First, the fitting expression for model 1 is given by $P(f) = p_0\sqrt{f} + p_1 f^2 + p_2 f^4$ where $p_0$, $p_1$ and $p_2$ are the unknown parameters. Table 4.4 recalls the observed power values $P$ values in Watts per processing frequency $f_{proc}$ in MHz or its normalized version.

| Model | Power($f$) | $r^2$ |
|---|---|---|
| 1 | $p_0\sqrt{f} + p_1 f^2 + p_2 f^4$ | 0.9984 |
| 2 | $p_0 f + p_1 f^2 + p_2 f^3 + p_3 f^4 + p_4 f^5 + p_5 f^6 + p_6 f^7$ | 0.9999 |
| 3 | $p_0 f log(f) + p_1 f + p_2 f^3 + p_3 f^5$ | 0.9916 |
| 4 | $p_0 f + p_1 f\sqrt{f} + p_3 f + p_3 f^5$ | 0.9986 |

**Table 4.3:** *Interpolation models of the power consumption as a function of the normalized frequency $f$*

| $f_{proc}$ (MHz) | 250 | 300 | 350 | 400 | 450 | 500 | 550 | 600 | 800 | 900 | 1000 | 1100 | 1200 | 1300 | 1400 | 1500 | 1600 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f$ | 0.15 | 0.18 | 0.21 | 0.25 | 0.28 | 0.31 | 0.34 | 0.37 | 0.50 | 0.56 | 0.62 | 0.68 | 0.75 | 0.81 | 0.87 | 0.93 | 1 |
| $P$ (Watts) | 0.27 | 0.30 | 0.32 | 0.37 | 0.44 | 0.52 | 0.61 | 0.72 | 1.44 | 1.71 | 2.03 | 2.39 | 2.74 | 3.20 | 3.75 | 4.40 | 5.03 |

**Table 4.4:** *Power measurements $P$ in Watts per processing frequency*

For model 1, the generalized model of Equation (4.12) becomes the system of 17 equations expressed by Equation (4.16). Vector $\boldsymbol{y}$ is replaced by the power values of Table 4.4 and Matrix $\boldsymbol{X}$ is computed by combining the model 1 expression with the actual normalized frequency values $f$ of Table 4.4.

$$\begin{aligned} 0.27 &= p_0\sqrt{0.15} + p_1 0.15^2 + p_2 0.15^4 \\ 0.30 &= p_0\sqrt{0.18} + p_1 0.18^2 + p_2 0.18^4 \\ 0.32 &= p_0\sqrt{0.21} + p_1 0.21^2 + p_2 0.21^4 \\ &.. \\ 5.03 &= p_0\sqrt{1.00} + p_1 1.00^2 + p_2 1.00^4 \end{aligned} \tag{4.16}$$

Finally, $p_0$, $p_1$ and $p_2$ are obtained by applying the formula proposed in Equation (4.14). The same reasoning applies for the different models to find the regression coefficients of each of them and are reported in Table 4.5.

| Model | Coefficients value | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | p0 | p1 | p2 | p3 | p4 | p5 | p6 | p7 |
| 1 | 0.2680 | 4.4149 | 0.3378 | - | - | - | - | - |
| 2 | 44.3187 | -308.4337 | 1073 | -1958 | 1931 | -965 | 190 | -2.22 |
| 3 | 1.0197 | 2.9128 | 1.7585 | 0.3528 | - | - | - | - |
| 4 | 0.2029 | 6.7430 | -2.5781 | 0.6448 | - | - | - | - |

**Table 4.5:** *Coefficients per interpolation model of 4.3*

Table 4.3 reports the quality evaluation of different models reflected by the coefficient of determination. These models are combinations of single convex functions. In Figure 4.14, the four models are compared with the actual power values. From a coefficient of determination point of view, model 2 is the best performer.

However, the analysis of the generated coefficients in Table 4.5 reveals that model 2 and model 4 infringe the rule $p_i \geq 0, \forall i$. Therefore, only models 1 and model 3 within the presented examples can be formulated into the convex optimizer to run simulations.

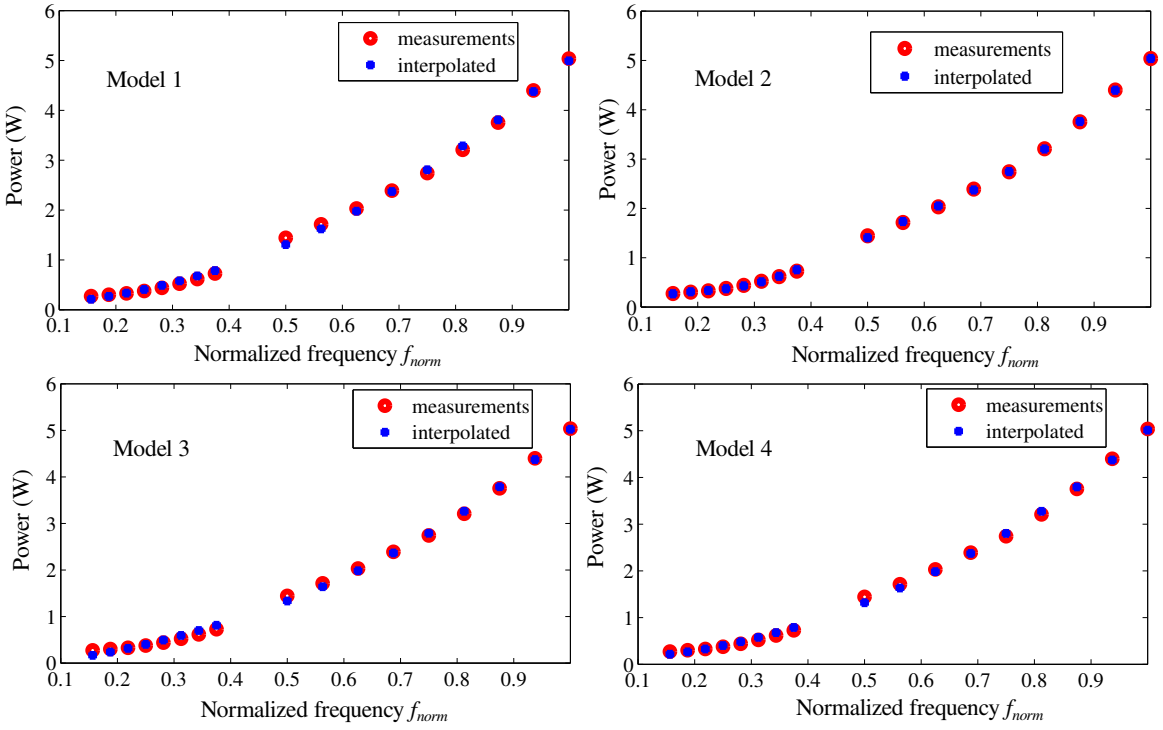


**Figure 4.14:** *Comparison of the four interpolation models versus the actual power measurements. $f$ is the normalized frequency to 1600 MHz*

Out of these two models, model 1 fits the best and is chosen to model the Exynos 5410 platform. Equation (4.17) gives the complete formulation of the power as a function of the normalized processing frequency.

$$P(f) = 0.268\sqrt{f} + 4.4149f^2 + 0.3378f^4 \tag{4.17}$$

This operation of platform characterization is necessary to obtain a mathematical expression of the power characteristics from the processing frequency. From the available formulations, only the convex ones are selected to the next step corresponding to the problem formulation.

### Final Problem Formulation

The final step of the framework proposed in Figure 4.13 consists in formulating the problem as a constrained problem. In Section 4.5.5, we determined a promising energy efficient point deriving it from the power measurements. Now that the power measurements can be formulated as a mathematical expression, the complete formulation of the optimization for the energy consumption can be done.

Let us consider model 1 of the previous section and formulate again Equation 4.10. It becomes Equation (4.18). It considers an application like the one given in Figure 4.8 composed on $N$ individual actors.

$$\begin{aligned} \underset{f_i}{\text{minimize}} \quad & \sum_{i=1}^{N} w_i \left(\frac{0.268}{\sqrt{f_i}} + 4.4149 f_i + 0.3378 f_i^3\right) \\ \text{subject to} \quad & \sum_{i=1}^{N} \frac{w_i}{f_{max} f_i} \leq D \\ & f_i \geq \frac{f_{min}}{f_{max}} \\ & f_i \leq 1 \end{aligned} \tag{4.18}$$

with $N$ the number of actors composing the application, $f_i$ the normalized processing frequency associated to actor $i$, $f_{max}$ (resp. $f_{min}$) 1.6 GHz (resp. 250 MHz), $w_i$ the load in cycles of actor $i$, $D$ the deadline of completion in seconds of the application.

*Disciplined Convex Programming* technique can be used because of the convexity properties of the formulation. All in all, the problem output is similar to the results presented before in Figure 4.10. When loosening the deadline, a most efficient set point appears which is not the slowest processing frequency, hence the *As-Slow-As-Possible* scheduling.

In the rest of this chapter, we propose to upgrade further the model to account with another capability of modern SoC: the multicore architecture.

## 4.4 Extension of the Energy Optimisation Problem to MPSoCs with DVFS and DPM

### 4.4.1 An Extension to Multicore Plaforms

Among the large set of possibilities offered by the Exynos 5410 platform, parallel processing model can be combined with the DVFS feature to design an energy efficient system. The parallelism of the architecture can be used intrinsically by the application with a multi-threaded execution. It is, for example, the case of the HEVC decoder discussed later in this chapter. In another field of application, the parallelism can be obtained by using an adequate transformation such as Single-Rate Data Flow (SRDF) as presented in our previous work [HNP+14].

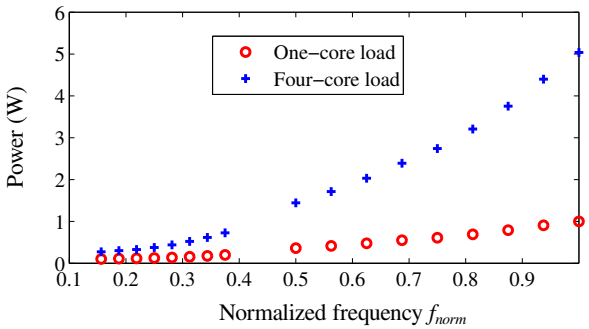


**Figure 4.15:** *Performance comparison between a one-core load and a four-core load*

The natural way to generate a general multicore model of the system energy consumption is to input the model with characteristics of platform loads on different numbers of cores. The `stress` benchmark can be configured to load the platform and perform power measurements, as shown in Listing 4.2.

```
# Stress on one core
stress --cpu 1
# Stress on four cores
stress --cpu 4
```

**Listing 4.2:** *Examples of stress benchmark calls*

As exhibited in Figure 4.15, the power consumption of the platform with one core loaded on the Exynos 5410 is very different from the power consumption when four cores are loaded. Although using fully loaded processors at their maximum speed can be very efficient from a real-time point of view, it shall be noted that it is not the case from the energy point of view. Indeed processing with four cores at their maximum speed needs more than four times a single core at its maximum speed.

As a consequence, the approach that assumes implicitly that processing in parallel is defacto the best strategy to design energy efficient systems need to be assessed thoroughly. Moreover, as shown in previous sections, parallel processing has a cost and is not perfect. This overhead needs to be accounted for when evaluating the multicore system energetic performance.

### 4.4.2 Multicore Energy Modeling with DVFS and DPM

The capability of processing in parallel is used to speed-up the application execution. For energy efficient design under real-time constraints, running *As-Fast-As-Possible* is not always a good solution. As depicted in Figure 3.4, the speed-up is not linear and reaches an asymptotic behavior. Therefore, using more speed-up may require more energy.

In this section, we upgrade our modeling by adding the parallelism level of the application into the overall optimization process. In the same manner as for the mono-core case, the energy characteristics are formulated as a function of the number of processing cores and the processing frequency: $E_{tot}(c_{proc}, f_{proc})$ with $f_{proc}$, the processing frequency and $c_{proc}$, the number of cores. These parameters can be either defined by their normalized or actual values. This model is justified by the fact that the overall power consumption is drastically modified as the number of active cores changes. Figure 4.15 exhibits this phenomenon on the Exynos MPSoC for one-core activation and a four-core activation.

A more complete picture is given in Figure 4.16. The average power power is depicted as a function of the number of core for the Exynos 5410 multicore platform. The configuration is given as per Table 4.2. The `stress` benchmark is run successively from one core to the maximum number of cores running at a time, i.e. four.

Section 4.3 establishes the relationship between the power and the energy per cycle of processing. With the same fashion, the energy model is built from power consumption measurements.

Figure 4.17 depicts the normalized energy after processing of the figure 4.16. It emphasizes the results of figure 4.12 and an energy efficient valley appears for the Cortex-A7 cluster. It also shows that using one core consumes less than using four cores. However finding the most efficient point to process given applications is not obvious and using one of the four corners is not the best solution.

#### Formulation of the Energy Optimization Problem in the Case of MPSoCs

Taking into account the parallelism, the optimization problem formulated in Equation (4.9) becomes:

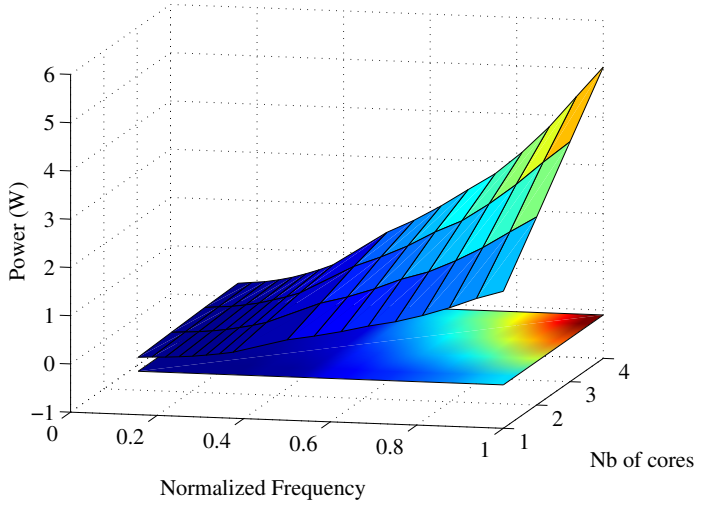


**Figure 4.16:** *System power trend with a N-core load*

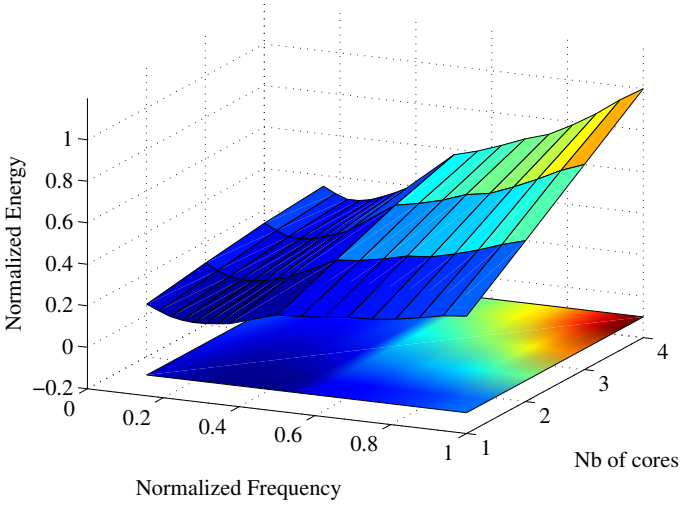


**Figure 4.17:** *Normalized energy as a function of the number of active cores and processing frequency*

$$\begin{aligned} \underset{f_{proc},c_{proc}}{\text{minimize}} \quad & E_{tot}(f_{proc}, c_{proc}) \\ \text{subject to} \quad & \frac{w(c_{proc})}{f_{proc}} \leq D \\ & f_{proc} \geq f_{min},\ f_{proc} \leq f_{max} \\ & c_{proc} \geq c_{min},\ c_{proc} \leq c_{max} \end{aligned} \tag{4.19}$$

with $f_{proc}$ the processing frequency in Hz, $w(c_{proc})$ the total load in cycles of the application after parallelization on $c_{proc}$ processing cores, $D$ the maximum time allowed to execute the applications, $f_{min}$ the minimum processing frequency in Hz supported by the platform, $f_{max}$ the maximum processing frequency in Hz supported by the platform, $c_{min}$ the minimum number of cores and $c_{max}$ the maximum number of cores.

Making the application truly run in parallel is not transparent. It reaches an asymptotic behavior when the number of cores grows. Therefore, it is proposed to integrate this trend into the optimization model of Equation (4.19) together with the energy per cycle expression.

For illustration purposes, the square root function is used to simulate the speed-up as function of the number of cores as shown in Figure 4.33.

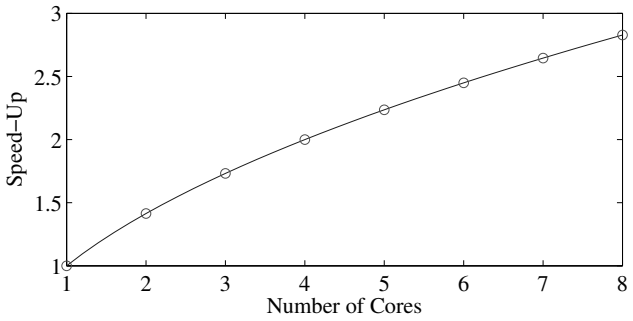


**Figure 4.18:** *Example of speed-up as a function of the number of cores*

The optimization problem becomes:

$$\begin{aligned} \underset{f_{proc},c_{proc}}{\text{minimize}} \quad & \frac{w}{\sqrt{c_{proc}}} E_{cycle}(f_{proc}, c_{proc}) \\ \text{subject to} \quad & \frac{w}{\sqrt{c_{proc}} f_{proc}} \leq D \\ & f_{proc} \geq f_{min},\ f_{proc} \leq f_{max} \\ & c_{proc} \geq c_{min},\ c_{proc} \leq c_{max} \end{aligned} \tag{4.20}$$

with $c_{proc}$ the number of processing cores $\in [c_{min}..c_{max}]$, $f_{proc}$ the processing frequency $\in [f_{min}..f_{max}]$, $D$ the deadline to execute the application and $w$ the actual load of the application in cycles.
The $E_{cycle}(f_{proc}, c_{proc})$ is directly extracted from the platform characteristics of Figure 4.17.

### 4.4.3 Extension of the Convex Solver Framework to Solve the MPSoC Energy Optimization Problem

Moving to MPSoC means parallel mapping and scheduling. But adding the parallelism as a parameter in the optimization process moves the problem to a multidimensional problem. As such, the *Disciplined Convex Programming* technique that is used in our framework of Section 4.3.1 cannot be exploited directly. Indeed, from the properties expressed in Section 4.3.1, the product of convex functions is not convex if no particular care is taken.

It is proposed in this Section to reformulate the problem to remove the unwanted products. If the problem is transformed with logarithm function, then it becomes an affine function problem where the products are replaced by sums. Boyd *et al.* [BKVH07] propose a solver for this class of problem. It is called Geometric Programming (GP) [Lue97]. GP is an optimization model where the variables are non-negative, and the objective and constraints are sums of powers of those variables with positive weights. Such a representation is also called a *posynom*.

A generalized *posynomial* function is any function obtained from a *monomial* expression using addition, multiplication and raising to constant positive power. A *posynom* is a function of the form

$$f(x) = \sum_{k=1}^{K} p_k x_1^{a_{k1}} ... x_n^{a_{kn}} \tag{4.21}$$

where $p \in \mathbb{R}_+^K$ and $\boldsymbol{A} = (a_{kj}) \in \mathbb{R}^{K \times n}$.

In the proposed framework of Figure 4.13, the MMSE estimator is used to find the fitting curve from power measurements. The same principle is used here and the curve is now a surface of two parameters: the normalized frequency $f$ and the normalized number of cores $c$. The regression models are computed from Equation (4.22).

$$P(c, f) = \sum_{i=0}^{N} \sum_{j=0}^{M} p_{i,j} c^i f^j \tag{4.22}$$

with $c$ the normalized number of cores, $f$ the normalized frequency and $p_{i,j}$ the regression coefficients.

Figure 4.19 is the example for polynomial approximation of the third order, i.e. $N = M = 3$. The coefficient of determination $r^2$ is used to measure the accuracy of the approximation.

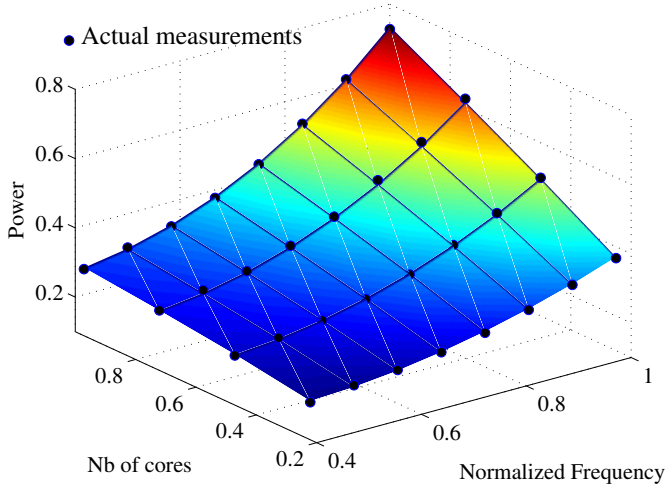


**Figure 4.19:** *Interpolated power consumption - Black dots: actual measurements*

Besides the model is not restricted to strict polynomial representation. Therefore reference *posynomial* functions are input into to the generation matrix as described in the proposed framework of Figure 4.13. The final goal is to input this reference model into the GP solver. Table 4.6 gives several model implementations.

The regression coefficients of each model are given in Table 4.7.

Model 3 and Model 4 have the property of using only polynomial strictly positive coefficients, hence they fulfill the conditions of a *posynomial* function expressed in Equation (4.21).

To generate the energy characteristics of the platform, the power model function is used of Table 4.6 and Equation (4.8) is applied.

For Model 3, the energy per cycle count is given by Equation (4.23).

$$E_{cycle}(c, f) = p_0 \frac{1}{f} + p_1 \sqrt{f} c^{1.5} + p_2 + p_3 f^2 c + p_4 f^6 c \tag{4.23}$$

| Model | Power, $P(f,c)$ | $r^2$ |
|---|---|---|
| 1 | $\sum_{i=0}^{N}\sum_{j=0}^{M} p_{i,j}c^i f^j$, M=2,N=2 | 0.999411 |
| 2 | $\sum_{i=0}^{N}\sum_{j=0}^{M} p_{i,j}c^i f^j$, M=1,N=1 | 0.958681 |
| 3 | $p_0 + p_1 f^{1.5}c^{1.5} + p_2 f + p_3 f^3 c + p_4 f^7 c$ | 0.990947 |
| 4 | $p_0 + p_1\sqrt{f}\sqrt{c} + p_2 f + p_3 f^3 c + p_4 f^7 c$ | 0.997025 |

**Table 4.6:** *Coefficient of Determination w.r.t of the interpolation model*

| Model | $p_{i,j}$ |
|---|---|
| 1 | 0.153 0.413 -0.150 -0.059 -1.083 0.34 0.117 1.314 -0.325 |
| 2 | 0.099 -0.196 0.1417 0.6377 |
| 3 | 0.138 0.1464 0.1001 0.273 0.0791 |
| 4 | 0.0313 0.2057 0.0815 0.2515 0.1242 |

**Table 4.7:** *Regression coefficients values*

For Model 4, the energy per cycle count is given by Equation (4.24).

$$E_{cycle}(c,f) = p_0\frac{1}{f} + p_1\frac{1}{\sqrt{f}}\sqrt{c} + p_2 + p_3 f^2 c + p_4 f^6 c \tag{4.24}$$

with $p_i$ issued from Table 4.7. Figure 4.20 shows the actual measurements and the generated energy values. These models are used in next sections for energy optimization purposes.

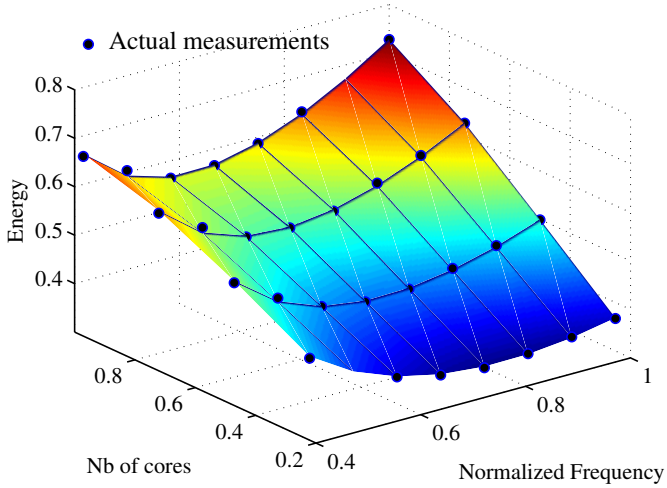


**Figure 4.20:** *Interpolated energy characteristics (nJ/cycle) as a function of the number of active cores (normalized) and the processing frequency (normalized) - Black dots: actual measurements*

### 4.4.4 Exploiting the MPSoC Energy Model to Explore the Design Space for Signal Processing Applications

Signal processing applications are often represented at a top level with static dataflow graphs for their expressiveness properties [Des14]. Information like workload $w$ per actor

of processing block can be given and is used to perform offline analysis. It is, therefore, an efficient tool to make early design tradeoffs and verification of the throughput requirements. Figure 4.21 is an illustration of application representation.

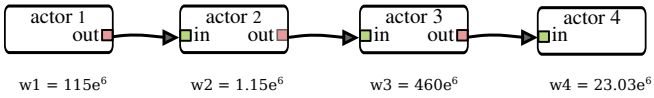


**Figure 4.21:** *Example of a dataflow application*

With the fixed number of instructions to be executed before hitting a deadline, it is possible to compute the appropriate processing frequency $f_{proc}$ for each actor of the graph. $\boldsymbol{f}$ is an N-length vector where each element $f_i$ corresponds to the normalized processing frequency of actor $i$. On top of that, signal processing applications can usually be processed in a parallel fashion. This parallelism is used to improve the throughput of the application for example. Similarly to the processing frequency, $\boldsymbol{c}$ is an N-length vector where each element $c_i$ corresponds to the normalized number of active cores (level of parallelism) for actor $i$.

When it comes to energy efficiency, these parameters $f_i$ and $c_i$ can be optimized as long as the real-time performance is guaranteed. The proposed framework exploits the intrinsic parallelism of the application together with the frequency scaling to minimize the energy consumption while meeting the real-time requirements.

### Models of Power and Speed-Up

The framework gathers both the platform and application characteristics with an analytic point of view. To get a full picture of the different use cases, it is proposed to model two types of platform. They are both issued from Model 3 of Section 4.4.3. We slightly change the coefficient $p_0$ to analyze the effect of static power. Sub-model 1 has a small leakage with $p_0 = 0.138$ and for Sub-model 2 has 0.045 W more of leakage with $p_0 = 0.183$. As the maximum dissipated power is 0.78 W, it represents 5% of the total. Figure 4.22 and Figure 4.23 depict the characteristics of both sub-models.

We also consider the following set of operating frequencies per actor $i$, $f_i$, $\{100...600\}$, $\in \mathbb{R}$ in MHz. The MPSoC is composed on four processors with $c_i$, $\{0.25..1\}$, $\in \mathbb{R}$.

At the application level, two types of behaviors are selected. The first one assumes a perfect behavior and scales perfectly when the number grows. The second one considers a not-perfect behavior with the following trend $SU = k_0.c^{0.25}$, with $SU$ the achieved speed-up, $k_0 = 2.0$, $c$ the normalized allocated cores. This model is issued from video applications that are further depicted in Section 4.5.4 in this chapter. Figure 4.24 compares the two schemes.

### Scheduling and mapping

When scheduling on a MPSoC a real-time application, designers are often to choose either an *As-Fast-As-Possible* scheduling or an *As-Slow-As-Possible* scheduling. *As-Fast-as-Possible* is easy to define as it consists in using the maximum of available cores at the maximum processing frequency. Figure 4.25 presents an example of *As-Fast-As-possible* scheduling on four cores. Each actor is split in four and runs at the maximum frequency.

*As-Slow-As-Possible* scheduling scheduling is more ambiguous as there are many combinations that can lead to finish the application just in time. Here, we propose to choose

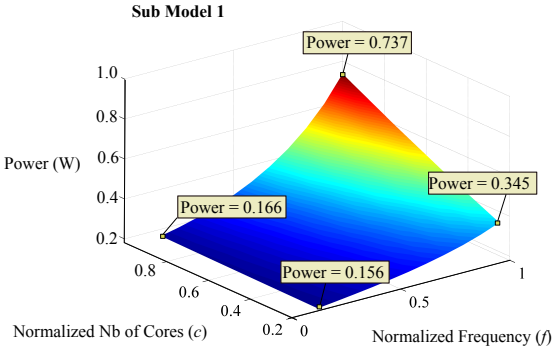


**Figure 4.22:** *Sub-Model 1: Power characteristics*

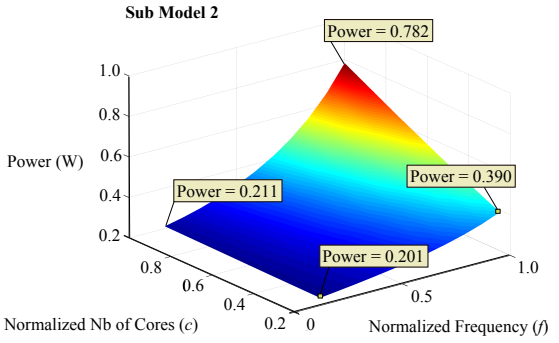


**Figure 4.23:** *Sub-Model 2: Power characteristics*

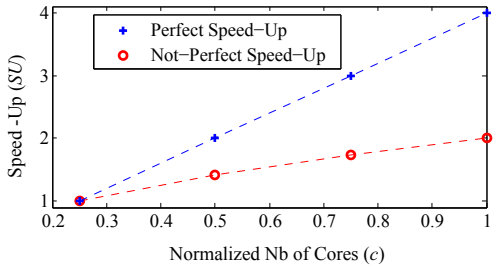


**Figure 4.24:** *Speed-up models - perfect and not perfect - as a function of the core usage*

*As-Slow-As-Possible* scheduling that performs load balancing over the available cores and minimizes the average frequency use. This approach is commonly used on MPSoC [SKH95].

Figure 4.26 presents an example of *As-Slow-As-Possible* scheduling on four cores. Each actor is also split onto the four available cores (load balancing) and the frequency is scaled per core so as to make the execution time match with the deadline.

The third approach uses the proposed framework to find the adequate scheduling. On top of respecting the real-time constraint, it adds energy efficiency in the middle of the optimization process. The cores can use frequency scaling and the actors can be scaled on one to four cores. Figure 4.27 shows an example of realization of the proposed scheduling.

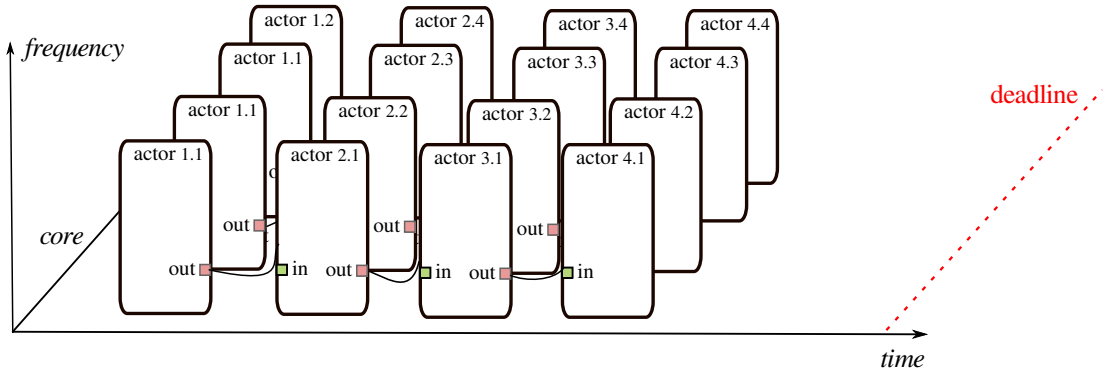


**Figure 4.25:** *Example of realization of As-Fast-As-Possible scheduling*

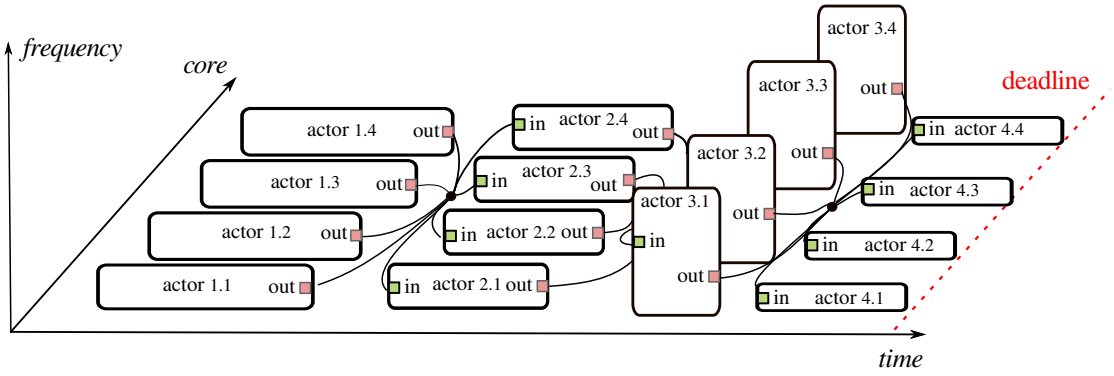


**Figure 4.26:** *Example of realization of As-Slow-As-Possible scheduling*

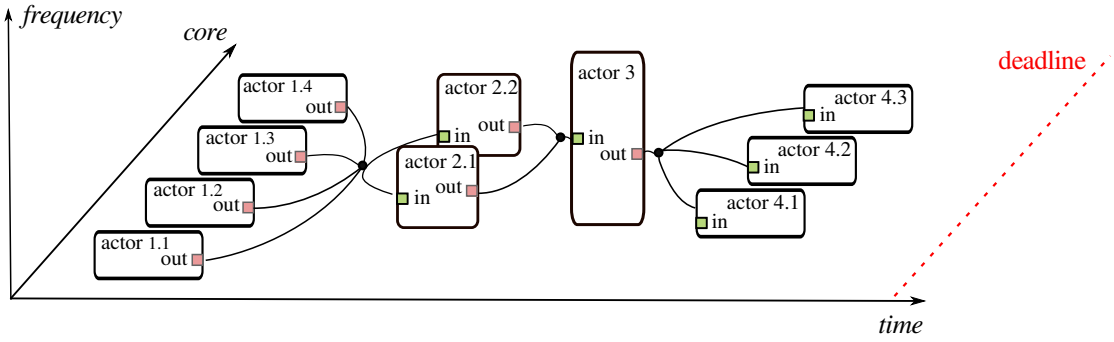


**Figure 4.27:** *Example of realization of the proposed scheduling. The number of actors used can vary as well as the frequency scaling*

However there are two variables to compute: the processing frequency per actor $i$ and the number of cores to use per actor $i$. The optimization problem is formulated in Equation (4.25) for the *not perfect* speed-up case.

$$
\begin{aligned}
\underset{f_i, c_i}{\text{minimize}} \quad & \sum_{i=1}^{N} \frac{L_i}{k_0\, {c_i}^{0.25}} \left(p_0 \frac{1}{f_i} + p_1 \sqrt{f_i} c_i^{1.5} + p_2 + p_3 f_i^2 c_i + p_4 f_i^6 c_i\right) \\
\text{subject to} \quad & \sum_{i=1}^{N} \frac{L_i}{k_0\, {c_i}^{0.25}} \cdot \frac{1}{f_i f_{max}} \leq D \\
& f_i \geq \frac{f_{min}}{f_{max}},\ f_i \leq 1 \\
& c_i \geq \frac{c_{min}}{c_{max}},\ c_i \leq 1
\end{aligned}
\tag{4.25}
$$

with $N$ the number of actors composing the application (e.g $N = 4$ in Figure 4.21, $L_i$ the load per actor, $k_0$ the speed-up coefficient as defined in Figure 4.24, $f_i \in \mathbb{R}$ the normalized processing frequency per actor, $c_i \in \mathbb{R}$ the normalized number of allocated cores, $p_{0..4}$ the coefficients of the power model, $D$ the deadline for application completion in seconds, $f_{min}$ is the minimum processing frequency (e.g. 100 MHz), $f_{max}$ is the maximum processing frequency (e.g. 600 MHz), $c_{min}$ is the minimum number of cores (e.g. 1), $c_{max}$ is the maximum number of cores (e.g. 4).
The problem is solved with GP programming technique as described in the previous section.

### Experimental Set-up and Results

The graph described in Figure 4.21 is considered for the performance evaluation. It is assumed to be real-time and shall finish before hitting a deadline. Each actor is characterized by a given load and the total load is $w_{tot} = \sum_{i=1}^{4} w_i = 6.10^6$ cycles. So one iteration run at the maximum frequency on one core takes 1 second.

Three different deadlines are tested hereafter: 0.5 second, 1.0 second and 1.5 seconds with a perfect scaling of the processing on the four cores or using an not-perfect rule as depicted in Figure 4.24. The time of execution is measured as well as the overall energy after an iteration execution.

#### *Results of Power Sub-model 1 - Model dominated by dynamic power*

Table 4.8 lists the performance achieved by the proposed method compared to *As-Slow-As-Possible* and *As-Fast-As-Possible* scheduling methods. First, it shows that choosing the best scheduling from the straightforward approaches, *As-Slow-As-Possible* and *As-Fast-As-Possible*. Gains from 5% to 27% can be observed depending on the deadline requirement and the speed-up model. For example, for Deadline=0.5, *As-Slow-As-possible* is more efficient that *As-Fast-As-Possible* for the perfect speed-up. However, this trend is opposite for Deadline=1.0. In all cases, our proposal achieves the most energy efficient scheduling. With this model, it also shows that the general trend is the following: reducing the processing frequency is more efficient that putting the cores into deep sleep state. Gains depend on the deadline requirement and the speed-up model.

| **Deadline=0.5** | Time (s) | $f$ | $c$ | Energy(J\|norm) |
|---|---|---|---|---|
| | | Perfect Speed-Up | | |
| Proposed | 0.44 | [0.56 0.56 0.56 0.56] | [1.00 1.00 1.00 1.00] | **0.1976**\|1.00 |
| ASAP | 0.50 | [0.41 0.41 0.53 0.41] | [1.00 1.00 1.00 1.00] | 0.2076\|1.05 |
| AFAP | 0.25 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.2191\|1.10 |
| | | Not-Perfect Speed-Up | | |
| Proposed | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | **0.4381**\|1.0 |
| ASAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | **0.4381**\|1.0 |
| AFAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | **0.4381**\|1.0 |
| **Deadline=1.0** | | | | |
| | | Perfect Speed-Up | | |
| Proposed | 0.44 | [0.56 0.56 0.56 0.56] | [1.00 1.00 1.00 1.00] | **0.1976**\|1.00 |
| ASAP | 0.60 | [0.41 0.41 0.41 0.41] | [1.00 1.00 1.00 1.00] | 0.2269\|1.19 |
| AFAP | 0.25 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.2191\|1.10 |
| | | Not-Perfect Speed-Up | | |
| Proposed | 0.82 | **[0.75 0.75 0.75 0.75]** | **[0.41 0.41 0.41 0.41]** | **0.3563**\|1.00 |
| ASAP | 1.00 | [0.41 0.41 0.53 0.41] | [1.00 1.00 1.00 1.00] | 0.4152\|1.16 |
| AFAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.4381\|1.23 |
| **Deadline=1.5** | | | | |
| | | Perfect Speed-Up | | |
| Proposed | 0.44 | [0.56 0.56 0.56 0.56] | [1.00 1.00 1.00 1.00] | **0.1976**\|1.00 |
| ASAP | 0.60 | [0.41 0.41 0.41 0.41] | [1.00 1.00 1.00 1.00] | 0.2269\|1.19 |
| AFAP | 0.25 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.2191\|1.10 |
| | | Not-Perfect Speed-Up | | |
| Proposed | 0.82 | **[0.75 0.75 0.75 0.75]** | **[0.41 0.41 0.41 0.41]** | **0.3563**\|1.00 |
| ASAP | 1,20 | [0,41 0,41 0,41 0,41] | [1.00 1.00 1.00 1.00] | 0,4538\|1.27 |
| AFAP | 0,50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0,4381\|1.23 |

**Table 4.8:** *Power sub-model 1: Energy consumption comparison of the Proposed method, As-Slow-As-Possible (ASAP) and As-Fast-As-Possible (AFAP) for deadline position at 0.5, 1.0 and 1.5 seconds on two models of speed-up (perfect and not perfect (defined in Figure 4.24). Evaluated metrics are the execution time in seconds and the energy. $f$ and $c$ are the normalized frequencies and core use per actor.*

### *Results of Power Sub-model 2 - Model dominated by static power*

Similarly to the previous section, Table 4.9 lists for sub-model 2 that suffers from static power. With this model too, our proposal achieves the most energy efficient scheduling. Gains from 5% to 34% can be observed depending on the deadline requirement and the speed-up model. However, in this case, it preferable to put the cores into deep sleep mode than using scaling frequency mechanisms. Therefore, as second best scheduling, using an *As-Fast-As-Possible* approach is better than using an *As-Slow-As-Possible* approach. It confirms that leakage power consumption is critical to define efficient scheduling. With our proposed modeling, the problem can be addressed for any a configuration.

### *Results Analysis*

When analyzing the power characteristics of the two proposed sub-models as depicted in Figure 4.22 and 4.23, one can argue that they are not dramatically different and it is difficult to identify the best operating point for energy efficient design. However, simulation results in Table 4.8 and Table 4.9 show that these power characteristics greatly influence the overall performance of the system. If the typical scheduling schemes - *As-Slow-As-*

| **Deadline=0.5** | Time (s) | $f$ | $c$ | Energy (J\|norm) |
|---|---|---|---|---|
| | | Perfect Speed-Up | | |
| Proposed | 0.40 | [0.62 0.62 0.62 0.62] | [1.00 1.00 1.00 1.00] | **0.2292**\|1.00 |
| ASAP | 0.50 | [0.41 0.41 0.53 0.41] | [1.00 1.00 1.00 1.00] | 0.2526\|1.10 |
| AFAP | 0.25 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.2416\|1.05 |
| | | Not-Perfect Speed-Up | | |
| Proposed | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | **0.4831**\|1.00 |
| ASAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | **0.4831**\|1.00 |
| AFAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | **0.4831**\|1.00 |
| **Deadline=1.0** | | | | |
| | | Perfect Speed-Up | | |
| Proposed | 0.40 | [0.62 0.62 0.62 0.62] | [1.00 1.00 1.00 1.00] | **0.2292**\|1.00 |
| ASAP | 0.60 | [0.41 0.41 0.41 0.41] | [1.00 1.00 1.00 1.00] | 0.2809\|1.22 |
| AFAP | 0.25 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.2416\|1.05 |
| | | Not-Perfect Speed-Up | | |
| Proposed | 0.76 | **[0.81 0.81 0.81 0.81]** | **[0.40 0.40 0.40 0.40]** | **0.4178**\|1.00 |
| ASAP | 1.00 | [0.41 0.41 0.53 0.41] | [1.00 1.00 1.00 1.00] | 0.5052\|1.21 |
| AFAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.4831\|1.15 |
| **Deadline=1.5** | | | | |
| | | Perfect Speed-Up | | |
| Proposed | 0.40 | [0.62 0.62 0.62 0.62] | [1.00 1.00 1.00 1.00] | **0.2292**\|1.00 |
| ASAP | 0.60 | [0.41 0.41 0.41 0.41] | [1.00 1.00 1.00 1.00] | 0.2809\|1.22 |
| AFAP | 0.25 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.2416\|1.05 |
| | | Not-Perfect | | |
| Proposed | 0.76 | **[0.81 0.81 0.81 0.81]** | **[0.40 0.40 0.40 0.40]** | **0.4178**\|1.00 |
| ASAP | 1.20 | [0.41 0.41 0.41 0.41] | [1.00 1.00 1.00 1.00] | 0.5618\|1.34 |
| AFAP | 0.50 | [1.00 1.00 1.00 1.00] | [1.00 1.00 1.00 1.00] | 0.4831\|1.15 |

**Table 4.9:** *Power sub-model 2: Performance of the energy consumption of the Proposed method, As-Slow-As-Possible (ASAP) and As-Fast-As-Possible (AFAP) for deadline position at 0.5, 1.0 and 1.5 seconds on two models of speed-up (perfect and not perfect 4.24). Evaluated metrics are the execution time in seconds and the energy if $f$ and $c$ are the normalized frequencies and core use per actor.*

*Possible* and *As-Fast-As-Possible* - were to be compared, the trend would be to decide the most appropriate schemes based on the leakage power performance of the considered platform. This analysis is not trivial from power measurements data only.

Energy efficiency characteristics are computed from Equation (4.8). They offer more insights to identify operating points. They are depicted per sub-model in Figure 4.28 and Figure 4.29. It notably exhibits the loss of efficiency of the minimum processing frequency when static power arises (Figure 4.29).

Finally, using the framework that is introduced in this chapter, an energy efficient scheduling (so called *Proposed* in Table 4.8 and Table 4.9)can be defined that outperforms the typical ones. Whatever the type of deadlines (from tight to loose) and whatever the power model (static or dynamic oriented), a best operating point can be computed that respects the real-time requirements and is energy efficient. It finds the best tradeoff between scaling the processing frequency and scaling the processing onto several cores to achieve low energy scheduling.

The resource allocation of the *Proposed* scheme needs to be highlighted for **Deadline=1.0** as it is none of the remarkable operating point of the platform. It is highlighted in bold blue in Table 4.8 and Table 4.9.

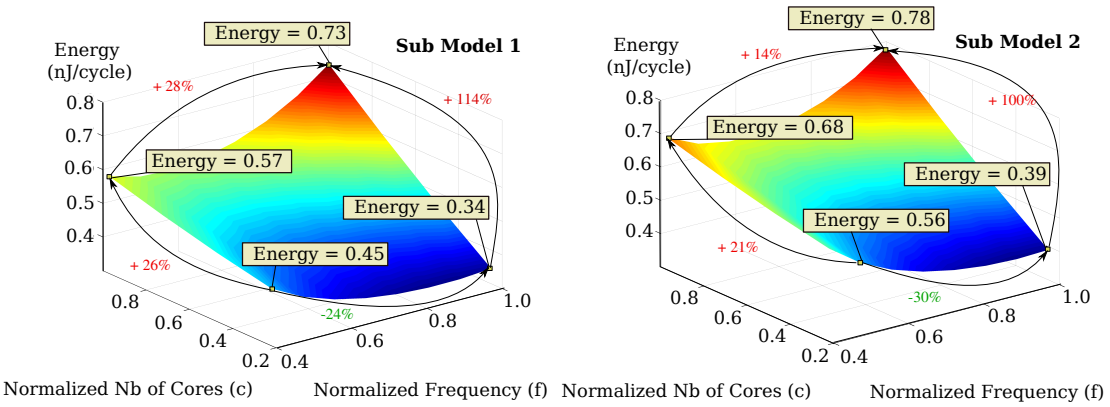


**Figure 4.28:** *Energy efficiency of Sub-Model 1* **Figure 4.29:** *Energy efficiency of Sub-Model 2*

### Discussions and Restrictions of the Model

The proposed model has the great advantage of finding the best match between the number of cores and the real time requirements. It shall be noted that it is flexible and enables an optimization at fine grain i.e per actor of processing. The time needed to solve the system can challenge previous work based on MILP and can be applied to more complex applications with more actors.

For illustration purposes, Figure 4.30 analyzes the solving time in seconds per number of actors composing the applications. The measurements are performed on a lap top equipped with processor Intel Core i5 using `CVX` tool [GBY08].

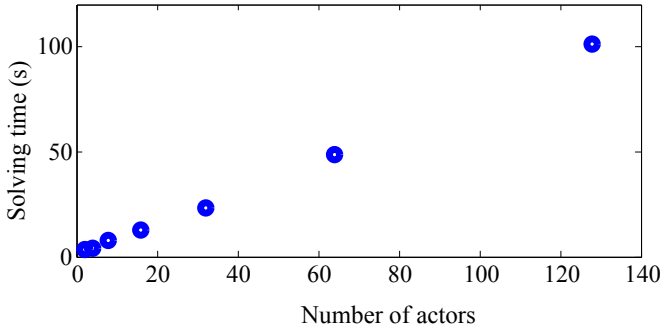


**Figure 4.30:** *Solving time using Geometric Programming with `CVX` per number of actors composing the application*

Besides, one can argue that the output of the proposed framework can be real values instead of integers. Especially for the number of active cores, it can be difficult to map onto fractional part of cores. As seen further in Section 4.5.3, this parameter can be finely tuned on MPSoC with the number of allocated threads. The number of allocated threads is then used as parameter to tune the number of active cores.

On top of that, it shall be highlighted that the model assumes the applications to optimize and the targeted platform to fulfil the following properties:

- the system is constrained by a deadline,
- each actor has a fixed workload within streaming-like application,
- the underlying platform supports dynamic frequency scaling and dynamic power management.

One can argue that not all signal processing applications can follow these properties. For example, video decoding is known to vary drastically from frame to frame. Therefore, a video decoding application needs to be considered at top level and, instead of analyzing the video at a frame level, the complete decoding is analyzed. This use case matches the offline decoding use case. The challenge is then to identify the best operating point from the design space.

The rest of this chapter shows how to use the proposed framework in the scope of offline video decoding and to find the best operating point.

## 4.5 Energy Efficient Offline Video Decoding: Use Case and Application to HEVC

In this section, the focus is put on the video decoding use case. Real-time video decoding is known to have a high processing load [NBP+15]. Therefore it infringes the rule that each processing actor has a fixed workload.

### 4.5.1 Context of an HEVC Decoder

Offline video decoding is a particular case of video decoding. The system is not constrained by a tight deadline and if a fluid play-back is needed, then a lag-time buffer is used. This section shows that the dominant part of a received video management energy is consumed by the decoding itself and not by the rendering part. If real-time is not demanded and if rendering can be delayed, then it makes sense to find the best energy efficient strategy to decode the sequence.

The challenge for the designer is to find the most appropriate processing point. Given a multicore platform, it means finding the best processing frequency in conjunction with the appropriate parallelism level.

This section focuses mainly on the offline decoding of a video content. It is used for example for video analytics, transcoding and simply for non-live playback. With the explosion of VOD services, this use case impacts a large range of devices. This case is illustrated by the experiment hereafter in Figure 4.31.

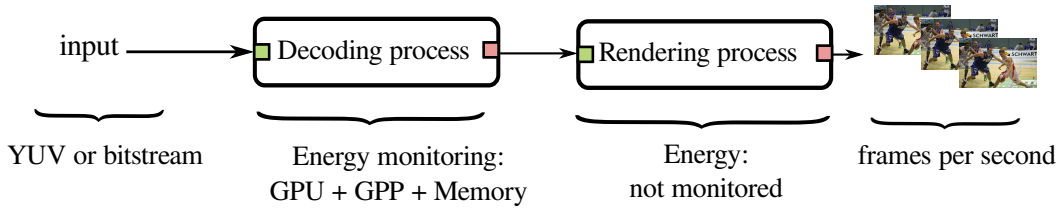


**Figure 4.31:** *Offline decoding set-up: input is either a compressed bitstream or decompressed YUV. The decoding is energy monitored and the rendering part is out of the scope.*

Our purpose is to compare the power consumption of two use cases. Firstly the decoding process handles only YUV frames. The decompression was performed offline first. Secondly the decoding process handles the decompression online. In both cases the playback is set to 24 fps and power consumption concerns the SoC only. The rendering part, namely the screen, is handled on a different power supply and is not monitored.

The Device-Under-Test (DUT) is based on the quadcore big.LITTLE that is used throughout this study. More details on platform were given in Section 4.3.2. We consider here two opposite bit rates based on the same resolution 1920x1080. *Parkscene QP37*

is a low quality sequence with low bit rate where *Parkscene QP22* is a high quality sequence with a high bit rate. Table 4.10 compare the power consumption of these two scenarios. Listing 4.3 provides the command line to play the video display using *ffmpeg* [Tom06].

```
# Play back of HEVC bitstream at 24 fps
ffplay -framerate 24 -autoexit ParkScene_1920x1080_24_qp37.hevc
# Play back of uncompressed HEVC bitstream at 24 fps
ffplay -vcodec rawvideo -s 1920x1080 -framerate 24 -autoexit
    ParkScene_1920x1080_qp37.yuv
```

**Listing 4.3:** *Video playback with ffmpeg*

The decompression produces a YUV sequence that can be displayed. As expected, the power consumption is not depending on the bit rate but on the resolution. Since the two sequences share the same resolution, the associated power consumption is similar if the system has to play uncompressed YUV input. However if decompression has to performed online, then the decoding process consumes more power and is the dominant part of the system consumption.

| | | Park Scene QP37 | Park Scene QP22 |
|---|---|---|---|
| Input | Compressed: HEVC bitstream | 0.5992 | 1.0579 |
| | Uncompressed: YUV | 0.2534 | 0.2524 |

**Table 4.10:** *Comparison of the power consumption between uncompressed and or compressed video playback.*

From an application point of view, it also means that it makes sense to optimize the decompression procedure first. If tight real-time is not demanded, then optimizing offline decoding can provide energy and power savings at system level. This section focuses on this use case and provides means to the designer to develop energy efficient systems.

### 4.5.2 Experimental Set-up

The same platform as described in Table 4.2 is used. The input videos are selected from the reference video test list of the BBC [bbc] and listed in Table 4.11. For each sequence, the two major profiles, Random Access (RA) and Imain (Imain), are tested.

| Class | Sequences | Resolution | Frame rate (Hz) |
|---|---|---|---|
| B | *Kimono (Km)* | 1920x1080 | 24 |
| | *ParkScene (P)* | | 24 |
| C | *BasketballDrill (BD)* | 832x480 | 50 |
| | *BQMall (BQM)* | | 60 |
| E | *KirstenAndSara (KS)* | 1280x720 | 60 |

**Table 4.11:** *Video sequences considered in the experiments*

### 4.5.3 HEVC Decoder Parallelism and Speed-up

The capability of decoding the video in parallel is a prime factor to accelerate the execution time. The *OpenHEVC* decoder can perform parallel decoding with multiple threads as described in Section 2.3.5. However, the actual parallelism cannot be parameterized directly. Besides, it does not equal the number of allocated threads. The section analyses how the thread allocation is used to tune the parallelism level $c$.

Listing 4.4 shows a command line example to decode using the number of threads as a example.

```
# HEVC decoding based on frame parallelism (-f 0) on 4 threads (-p 4)
./hevc mysequence.bin -f 0 -p 4
```

**Listing 4.4:** *Example of call Open HEVC call to decode mysequence.bin with 4 treads*

The thread allocation performance is analyzed in Figure 4.32. The input sequences of Table 4.11 are run onto the DUT described in Table 4.2. The parallelism level axis expresses how efficient the utilization of the four cores is. 100% of parallelism means the execution time is directly improved by four.

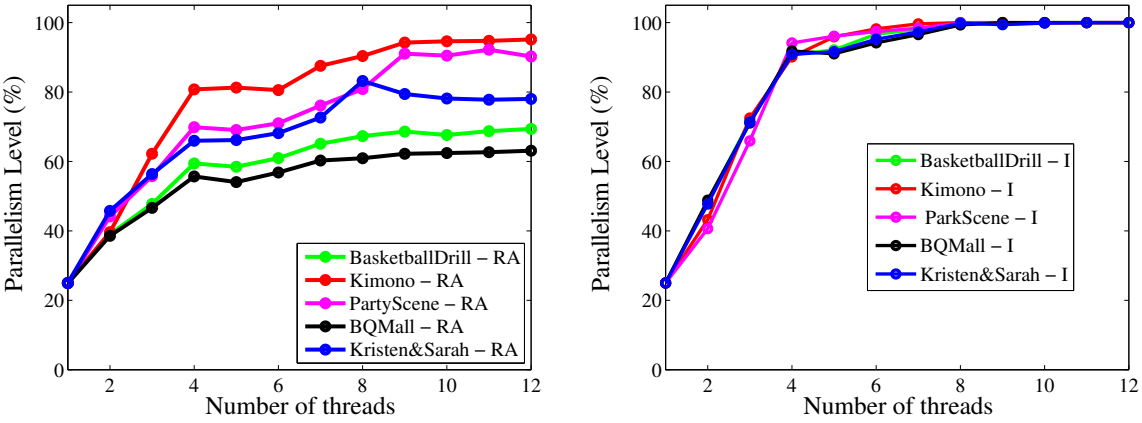


**Figure 4.32:** *Analysis of the parallelism level as a function of the thread allocation. Left is with Random Access profile sequences and Right is All Intra profile sequences*

It also shows that increasing the number of allocated threads improve the parallelism level, hence the execution time. However this behavior follows an asymptotic trend. It also varies with the encoding profile. Indeed the allocated threads and the parallelism have a close-to-linear behavior with All Intra (AI) encoding. Naturally it is independent from the sequence content. RA encoding shows different characteristics. The trend is content dependent and allocating more threads provide more speed-up but with an asymptotic behavior. It also provides a fine grain tuning of the number of active cores.

From an energy point of view, it means that speeding-up applications costs more when the level of parallelism is already high.

### 4.5.4 Determining the Most Energy Efficient Point at the Design Phase

The previous sections exhibit that there exists an energy efficient setting point. However, this setting point is determined after a first run of the video decoding.

Here, we propose to use the output of the generic benchmark that measure the power consumption together with the parallelism performance of the application. We can derive

from the power measurements an energy model that takes into account the parallelism inside the application. The best setting point is finally extracted after the optimization phase.

### Model of the HEVC Decoder Parallelism

The purpose of the parallelism model is to get a representation of the performance. As the trend can be asymptotic, it must be accounted into the energy budget. The trend of the speed-up ($SU$) is modeled by Equation (4.26) as a function the core usage ($c$) and shown in Figure 4.33. As there are four cores on the platform, four is the maximum speed-up that can be reached.

$$SU = k_0 \, c^{0.25} \tag{4.26}$$

with $k_0$ described in Table 4.12.

| Sequence | BQM | K | PS | BD |
|---|---|---|---|---|
| $k_0$ | 2.5052 | 3.6148 | 3.2998 | 2.7085 |

**Table 4.12:** *$k_0$ coefficient of Equation (4.26)*

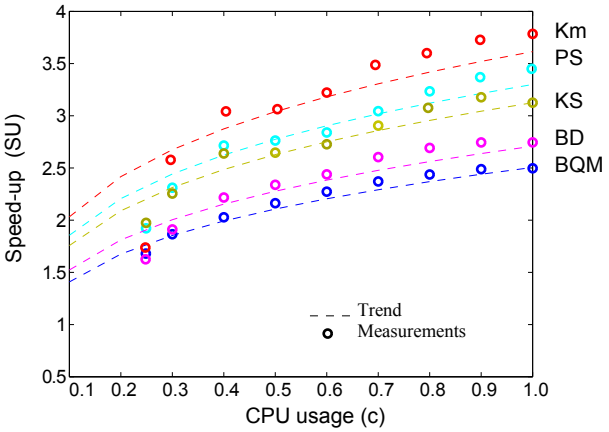


**Figure 4.33:** *Speed-up modeling from actual measurements from Section 4.5.3*

### Formulating the Offline HEVC Decoder Energy Optimization Problem

In a general context, the optimization problem is formulated in Equation (4.20). For the offline decoding use case, the problem is relaxed because there is no real-time requirement.

If the energy model is taken from the generic power formula in Equation (4.24) and the parallelism performance from the Section 4.5.4, then the overall optimization problem can be formulated as follows.

$$\begin{aligned} \underset{f,c}{\text{minimize}} \quad & \frac{L}{k_0\, c^{0.25}}\left(p_0\frac{1}{f} + p_1\sqrt{f}c^{1.5} + p_2 + p_3 f^2 c + p_4 f^6 c\right) \\ \text{subject to} \quad & f \geq \frac{f_{min}}{f_{max}}, f \leq 1 \\ & c \geq \frac{c_{min}}{c_{max}}, c \leq 1 \end{aligned} \tag{4.27}$$

with $L$ the actual load of the processing in cycles, $c$ the normalized number of cores, $c_{min}$ the minimum number of cores, $c_{max}$ the maximum number of cores, $f$ the normalized processing frequency, $f_{min}$ the minimum processing frequency, $f_{max}$ the maximum processing frequency, and $p_i$ the regression coefficients given in Table 4.13.

Table 4.13 recalls the coefficients of the real DUT from Equation (4.24).

| Coefficients | $p_{0..4}$ |
|---|---|
| Real Platform | 0.153 0.1464 0.1001 0.273 0.0791 |

**Table 4.13:** *Regression coefficients values*

### *Results of the rapid prototyping*

To solve the minimization problem, we use the GP solver presented in Section 4.4.3 on the two models. Table 4.14 summarizes the results for the different sequences.

| | Sequence | $f_{proc}$ (MHz) | $c_{proc}$ |
|---|---|---|---|
| Platform Model | K | 351 | 4 |
| | PS | 351 | 4 |
| | BQM | 351 | 4 |
| | BD | 351 | 4 |
| | KS | 351 | 4 |

**Table 4.14:** *Optimization results and setting point determination*

The results of the platform model can be confronted with the actual results of the video decoding on the DUT. The output of the rapid prototyping process states that the optimal configuration is to process the video at $f_{proc}$ set to 351 MHz and using the four cores. In Section 4.5.5, the same video sequences are thoroughly investigated with respect to the processing frequency and parallelism set with thread allocation.

The best setting parameters couple is $\{f_{proc}, p_{thread}\}$=$\{350$ MHz, $12\}$. They confirm the output of the rapid prototyping step as the processing frequency is very similar and the number of allocated threads is close to the one needed to achieve the maximum speed-up.

It shall be noted that the best setting point is not one of the remarkable point (mini/-maximum frequency, mini/maximum core utilization) of the design space. Using the proposed framework can ease the determination of this point at an early stage of the design.

### 4.5.5 Experimental Verification of the Offline Decoding Energy Optimal Setting Point

Defining the best processing point $\{f_{proc}, c_{proc}\}$ of an MPSoC is a challenging task. The previous section proposes to determine it from a rapid prototyping point of view. Indeed, the design space exploration is done buy solving the optimization problem.

In this section, we propose to check the results with an intensive exploration. All the possible operating points are tested. Also a comparison of the straight forward approach, that would consist in selecting one of the four *remarkable* couples, is provided: $\{f_{min}, c_{min}\}$, $\{f_{min}, c_{max}\}$, $\{f_{max}, c_{min}\}$, $\{f_{max}, c_{max}\}$.

The energy needed to decode the complete sequence is measured here. As the parallelism cannot be parameterized directly, the number of allocated threads is used instead as explained in Section 4.5.3. Two types of measurements are performed. The *Total energy* is the on-core energy (processors including the GPU) and off-core energy (memory). The *Core only* measurements excludes the consumption of the memory.

Figure 4.34 and Figure 4.35 report the normalized energy consumption for the Kimono and BasketBall Drill sequences of Table 4.11. Kimono varies slowly with a traveling where BasketBall Drill embeds fast movements on a fixed shot.

Even though the sequences are different or the encoding profile is different, the trends are similar. The following information shall be highlighted:

- **the most energy efficient point** $\{f_{eff}, p_{eff}\}$ is none of the standard setting points $f_{min}$, $f_{max}$, $p_{min}$, $p_{max}$
- **the *Total energy* and the *Core energy*** are significantly different. It is especially true when the processing frequency is low.
- **Minimizing the execution time** by minimizing the processing frequency is not the most energy efficient strategy.

The next experiments extract the most energy efficient setting point per input sequence and compare it with the *remarkable* points. Table 4.17 reports the *Core energy* with RA encoding profile. Table 4.16 reports *Total energy* with RA encoding profile. Table 4.15 reports the *Core energy* with RA encoding profile. Table 4.18 reports *Core energy* with AI encoding profile. Finally, Table 4.18 reports *Total energy* with AI encoding profile.

These experiments confirm the general trend and position the most efficient point at $f_{proc}$=350 MHz with a number of threads $p_{treads}$ from 10 to 12 in most cases. It shall be noted that using more than 10 threads has no impact on the performance because the speed-up has reached its asymptotic as depicted in Figure 4.32.

Finally among the *remarkable* points, the most energy efficient point consists in using the maximum number of threads with the minimum frequency. Compared to this point, the most energy efficient point is better from 12% to 20% in terms of *Total Energy*.

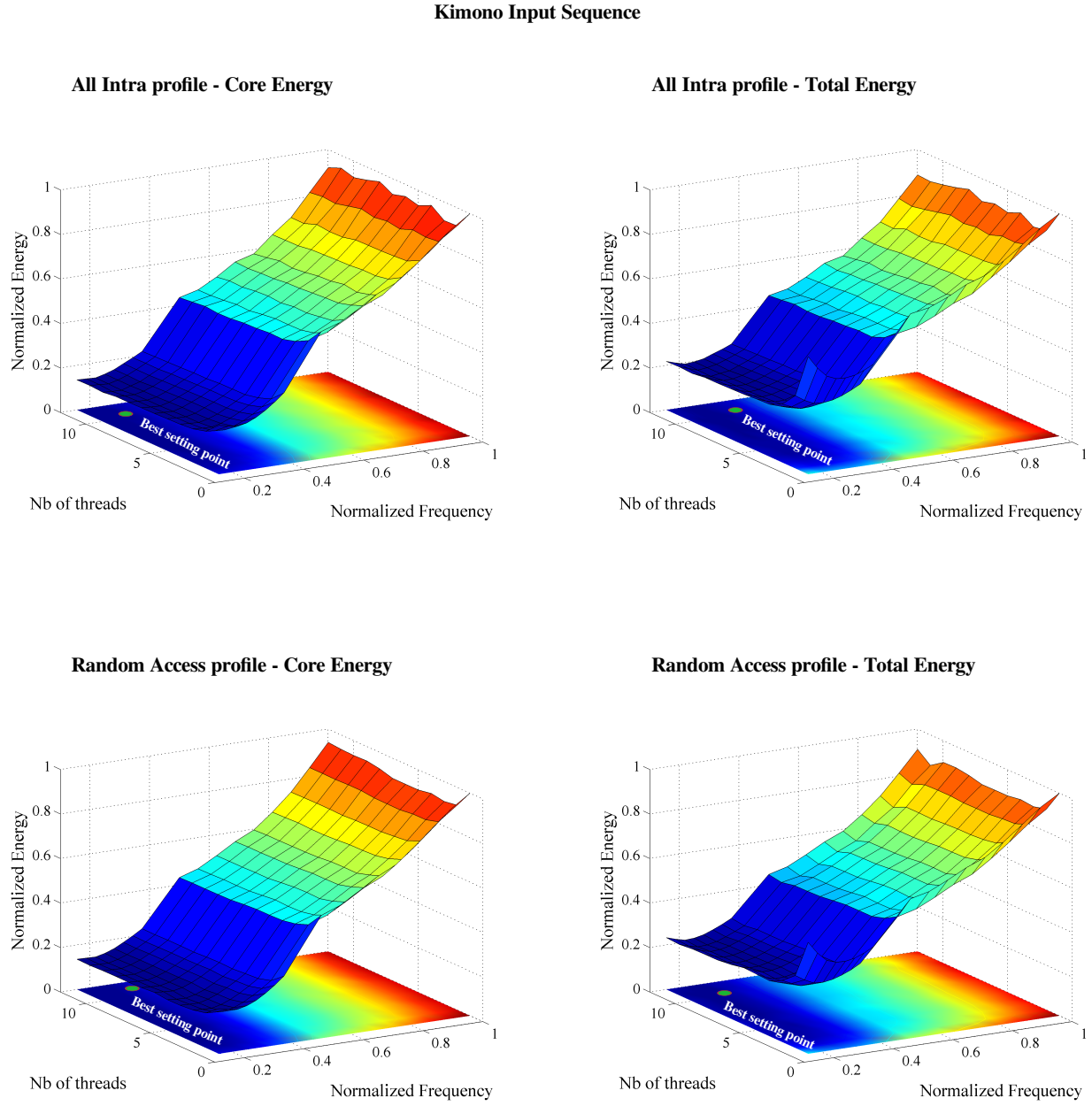


**Figure 4.34:** *Analysis of the energy efficiency to decode Kimono test sequence as a function of the processing frequency $f_{proc}$ and the number of processing threads $p_{thread}$. The most energy efficient point is for $f_{proc} = 350MHz$ with $p_{thread} = 10$ to 12. It confirms the operating point computed at the design phase $f_{proc} = 351MHz$ with a parallelism level of $c = 4$*

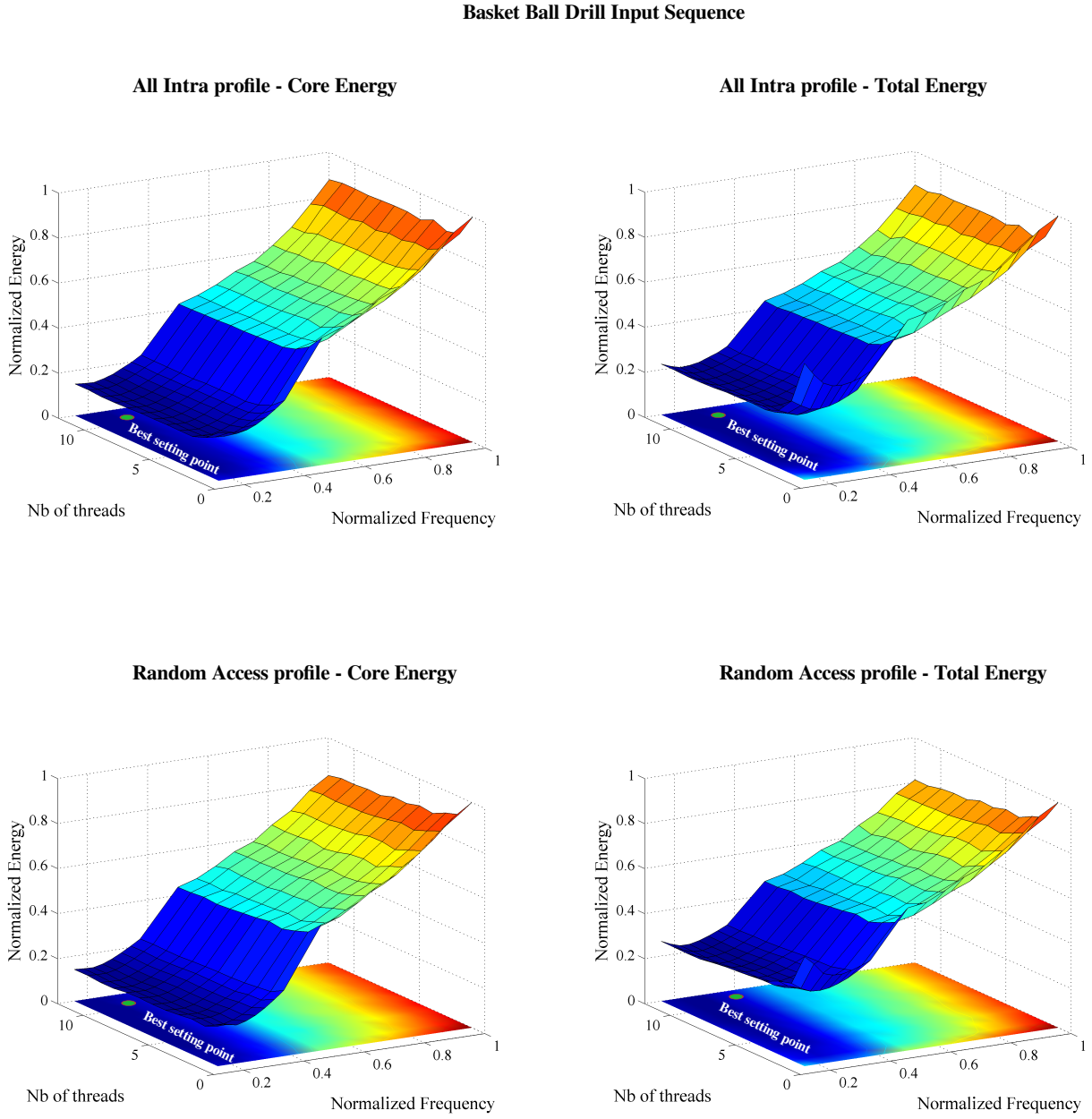


**Figure 4.35:** *Analysis of the energy efficiency to decode BasketBall Drill test sequence as a function of the processing frequency $f_{proc}$ and the number of processing threads $p_{thread}$. The most energy efficient point is for $f_{proc} = 350MHz$ with $p_{thread} = 10$ to 12. It confirms the operating point computed at the design phase $f_{proc} = 351MHz$ with a parallelism level of $c = 4$*

**Table 4.15:** *Random Access profile: Core Energy efficient point* $\{f_{eff}, p_{thread}\}$ *analysis and performance comparison with the typical points* $f_{min}$, $f_{max}$, $c_{min}$, $c_{max}$

| **Random Access profile – Core Energy** | | | | | | |
|---|---|---|---|---|---|---|
| Video | Optimal | | Gains (%) | | | |
| | $f_{proc}$ | $p_{thread}$ | $p_{min} - f_{min}$ | $p_{min} - f_{max}$ | $p_{max} - f_{min}$ | $p_{max}$ - $f_{max}$ |
| BD | 350 | 10 | 25.8 | 86.8 | 6.1 | 84.1 |
| BQM | 350 | 11 | 32.08 | 86.1 | 6.2 | 83.8 |
| KS | 350 | 12 | 33.5 | 87.6 | 12.7 | 85 |
| Km | 350 | 11 | 34.8 | 86.8 | 2.8 | 85.9 |
| P | 350 | 11 | 30.6 | 87.9 | 3.7 | 85 |

**Table 4.16:** *Random Access profile: Total Energy efficient point* $\{f_{eff}, p_{thread}\}$ *analysis and performance comparison with the typical points* $f_{min}$, $f_{max}$, $c_{min}$, $c_{max}$

| **Random Access profile – Total Energy** | | | | | | |
|---|---|---|---|---|---|---|
| Video | Optimal | | Gains (%) | | | |
| | $f_{proc}$ | $p_{thread}$ | $p_{min} - f_{min}$ | $p_{min} - f_{max}$ | $p_{max} - f_{min}$ | $p_{max}$ - $f_{max}$ |
| BD | 400 | 11 | 52.3 | 76.8 | 12.7 | 71.5 |
| BQM | 350 | 10 | 51.15 | 76.7 | 15.99 | 72.35 |
| KS | 350 | 12 | 56.5 | 78.9 | 20.1 | 73.8 |
| Km | 350 | 11 | 59 | 79.5 | 11.8 | 77.4 |
| P | 350 | 11 | 60.4 | 80.9 | 13.2 | 75.5 |

**Table 4.17:** *All Intral profile: Core Energy efficient point* $\{f_{eff}, p_{thread}\}$ *analysis and performance comparison with the typical points* $f_{min}$, $f_{max}$, $p_{min}$, $p_{max}$

| **All Intra profile – Core Energy** | | | | | | |
|---|---|---|---|---|---|---|
| Video | Optimal | | Gains (%) | | | |
| | $f_{proc}$ | $p_{thread}$ | $p_{min} - f_{min}$ | $p_{min} - f_{max}$ | $p_{max} - f_{min}$ | $p_{max}$ - $f_{max}$ |
| BD | 350 | 10 | 37.8 | 86.8 | 5.7 | 84.9 |
| BQM | 350 | 11 | 40.5 | 87.6 | 4.6 | 84.9 |
| KS | 350 | 12 | 39.7 | 87.4 | 3.1 | 84.9 |
| Km | 350 | 10 | 41.6 | 87.2 | 4.8 | 86.1 |
| P | 350 | 12 | 41.8 | 86.6 | 4 | 86.5 |

**Table 4.18:** *All Intral profile: Total Energy efficient point* $\{f_{eff}, p_{thread}\}$ *analysis and performance comparison with the typical points* $f_{min}$, $f_{max}$, $c_{min}$, $c_{max}$

| **All Intra profile – Total Energy** | | | | | | |
|---|---|---|---|---|---|---|
| Video | Optimal | | Gains (%) | | | |
| | $f_{proc}$ | $p_{thread}$ | $p_{min} - f_{min}$ | $p_{min} - f_{max}$ | $p_{max} - f_{min}$ | $p_{max}$ - $f_{max}$ |
| BD | 350 | 10 | 62.2 | 80.5 | 13 | 77 |
| BQM | 350 | 11 | 63.4 | 82.6 | 14.5 | 78.1 |
| KS | 350 | 12 | 61.4 | 81.5 | 11.2 | 77 |
| Km | 350 | 10 | 63.9 | 80.9 | 12.4 | 78.5 |
| P | 350 | 12 | 65.3 | 80.6 | 12.6 | 79.6 |

## 4.6 Conclusion and Perspectives

In this chapter, we propose:

- **an energy model** accounting for the capabilities of SoC to use DVFS and DPM. The model can be either applied to single core or to multicore platforms.
- **a rapid prototyping framework** gathering the generic application characteristics and the platform energy performance. The framework explores the design space on all the appropriate dimensions of the considered platform.
- **an optimization problem formulation** exploiting both the platform and the application characteristics to compute the most energy efficient operating point. Compared to other approaches based on heuristics, the problem resolution uses the convexity properties of the energy.

From a perspective viewpoint, the proposed framework can be utilized for a wide range of applications. From real-time applications on small things where the energy efficiency is crucial (e.g. Internet of Things area) to offline processing where processors performance can be easily tuned (e.g. offline processing on server), the framework can be used to help the designer to identify where to locate the operating point in the design space.

It assumes, though, that the application is static with a fixed workload. For future research, the following enhancements could be envisioned:

- **online tracking** of the energy efficient working point. Indeed, one can oppose that the power characteristics change with time and with thermal performance. To address this issue, the energy modeling could be managed online for platform embedding power sensors probes. As described in the proposed framework, the mathematical model is derived from measurements that could be performed online. Using MMSE as proposed to fit the measurements with the platform parameters is done after a matrix inversion. Good accuracy is obtained with $5 \times 5$ matrix to be inverted as shown in Section 4.3.2. This type of matrix inversion is commonly processed on embedded processors [NM14].
- **dynamic applications** could be addressed if the system is aided to determine the processing load and expand the scope of applications. Therefore a metadata could indicate the load of processing per application. For example, such mechanism is proposed to help video decoder within the GreenMetadata MPEG standard as described in Section 3.5.
- **conditional graph management** to take into consideration the energy. Indeed some graph transformations exist to make application as parallel as possible to optimize the load balancing. However, if real time is the primary concern, then we show that there can exist more energy efficient schemes than the maximum parallel ones.

In the next chapter, the case of dynamic signal processing applications and more precisely real-time HEVC decoding is studied.

CHAPTER 5

# Video Aware DVFS for Energy Efficient Video Decoding

## 5.1 Introduction

As presented in Chapter 4, high level representations combined with an adequate MoC is a powerful tool to explore the design space. Chapter 4 proposes in this scope a framework gathering the system requirements, the application characteristics and the platform low power features. This framework leads to the definition of an efficient rapid prototyping tool that defines a low energy consumption scheduling strategy. Nonetheless, the proposed framework relies on the a-priori knowledge of the application load per iteration of the algorithm and uses this knowledge to optimize the application energy consumption at compile time. It can not address applications that have a dynamic behavior like realtime video decoding. It is a real challenge to design a realtime video decoder that can provide the frames to display at a preset playback rate.

In this chapter, the focus is shifted to real-time HEVC decoding, which is a dynamic application. We investigate how power management techniques such as DVFS can be used online to minimize the overall power consumption of the system. The aim of this chapter is to show how the latest video standard can be implemented in realtime on GPP platforms with low power considerations. Software-based implementations can represent for HEVC the opportunity of a fast market development in consumer electronic devices. For devices that are already in the field, using HEVC hardware solutions is not possible and only software solutions can be envisioned to support HEVC. In this case, as seen in Chapter 3, many solutions are proven to be feasible. The cursor has now moved from feasibility to usability. Indeed the application shall guarantee a long up-time of execution accounting for low power performance.

This chapter is organized as follows. Section 5.2 recalls the different low power implementations from hardware to software-based solutions on GPP platforms. Section 5.3 exhibits the characteristics of the HEVC decoder. The targeted platform is described in Section 5.4 emphasizing the power managements capabilities like DVFS. Video-Aware HEVC decoders are proposed in Section 5.5 and power consumption as well as realtime performance are given in Section 5.6. Section 5.7 discusses the solution proposed in this chapter.

## 5.2 Existing HEVC Decoder Implementations

In this section, a review of the different available implementations of HEVC decoder is given. It focuses mainly on low power implementations.

### 5.2.1 Low Power HEVC Decoders

Hardware implementations of HEVC decoders have been recently proposed. Liu *et al.* [LCW+15] propose an HEVC decoder supporting 4K resolution at 60 fps. The energy consumption is given for a 28 nm CMOS technology. The clock frequency is set to 350 MHz. An energy consumption for the core around 0.2 nJ/px is reported. Hardware implementations offer the most energy efficient decoders and the highest performance. However, this performance comes at the expense of specializing the circuit to the application, leading to low flexibility and high development cost. Hardware implementations can easily be envisioned for new products. However, for the existing billions of devices, only software implementations could secure the future of a new standard like HEVC.

Recent multi and many-core architectures represent a serious opportunity to implement a realtime HEVC decoder. Recent work [CAML+13b] has demonstrated the ability to obtain a realtime software decoder on high resolutions up to 4K. This software implementation targets a laptop CPU with 4 cores from Intel, and a TILE-Gx36 with 36 cores from Tilera. These targets have moderate power consumption and are dedicated to high-performance computing. For these multi-core and many-core processors, task level parallelism is used to exploit WPP coding. Moreover, performance has been significantly improved compared to the reference HM code by integrating architecture independent optimizations as well as architecture dependent optimizations. Results are reported for Full HD and Ultra HD resolutions. For the 4-core Intel CPU, energy consumptions of 74 nJ/px and 86 nJ/px are obtained respectively for the Full HD and Ultra HD resolutions. For the 36-core Tilera CPU, an energy consumption of 161 nJ/px and 84 nJ/px are obtained respectively for the Full HD and Ultra HD resolutions.

### 5.2.2 DVFS-Based Video Decoders for GPP

With the ever growing capabilities of modern SoC, recent studies prove that multimedia applications like video can be done on the GPP sub-system while preserving realtime response. As a consequence, a pure software HEVC decoder can be designed for a GPP with a reasonable impact on the power consumption that is suitable to system requirements [CAMJ14, RNH+15]. Efficient decoding on a GPP leverages on optimized code implementations with SIMD acceleration and process parallelization. Additionally, the energy performance is analyzed as a function of the operating frequency. Chi *et al.* report that clear benefits can be obtained from frequency scaling [CAMJ14] but choosing the right operating frequency is a non-trivial task. The system shall not only be power efficient but shall also respect realtime requirements. In this work, we propose to use an on-line complexity actuator within the HEVC decoder: computation approximations, as well as frame complexity predictions based on frame type.

### 5.2.3 Dynamic Frequency Scaling Policies

In modern SoCs, two main power management techniques are provided that minimize the energy consumption. By combining clock gating and power gating, DPM [BD12] is used to turn a processing core into a low power state when it is not in use. To reduce the

influence of dynamic power, DVFS [KSY+02] reduces both the clock frequency and the supply voltage until the application realtime constraint is slightly exceeded. Each of these techniques impacts the total power. Equation (5.1) is the power expression at a system level [JPG04].

$$P_{tot} = V.I_{leak} + C_{eff}.f_{proc}.V^2, \tag{5.1}$$

where $V$ is the supply voltage corresponding to the processing frequency $f_{proc}$, $I_{leak}$ is the leakage current, and $C_{eff}$ is the circuit effective capacitance. The left-hand side of the expression is called *static power* and the right-hand side *dynamic power*.

The total power consumption is a convex function of the processing frequency as depicted in Figure 5.1. Hence, and by using the Jensen's inequality [HK82], Yao *et al.*[YDS95] propose, when executing two jobs, job 1 and job 2, to process both at the same frequency instead of adapting frequency to $f_1$ for job 1 and to $f_2$ for job 2. In other words, if we suppose that a job 1 shall be decoded at frequency $f_1$ and job 2 at frequency $f_2$, then the following inequality is obtained.

$$P\left(\frac{f_1 + f_2}{2}\right) \leq \frac{P(f_1) + P(f_2)}{2}, \tag{5.2}$$

Finally $f_1$ should be chosen equal to $f_2$ to minimize energy.

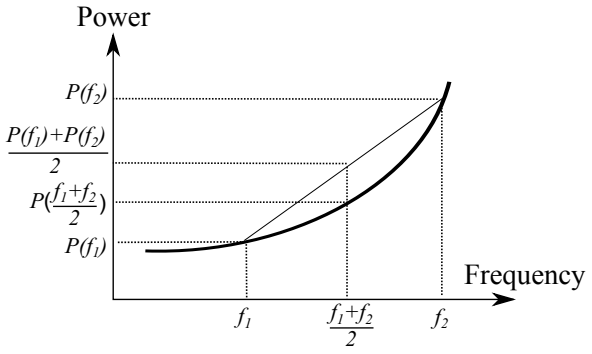


**Figure 5.1:** *Power as convex function of the frequency [BBS+15b]*

Therefore, Benmoussa *et al.* [BBS+15b] suggest that setting DVFS to the the average workload of a complete sequence is more efficient than using a per-frame DVFS. The simple example of Figure 5.2 illustrates this idea. Four frames are decoded and shall respect the deadline for delivery. When considering video decoding, the deadline is the inverse of the display rate of the video.

### 5.2.4 HEVC Decoders with Integrated DVFS

At an early stage of a circuit design, Gutnik *et al.* [GC97] propose to use dynamically adjustable power supplies as a method to lower power dissipation in DSPs. The study focuses on hardware implementations and how circuits can be designed with DVFS. In our approach, we propose to use an existing hardware and design a software set-up that is put on top and makes the most of the existing hardware resources.

Considering multimedia applications and video processing, Lu *et al.* [LLSS03] propose to use a frame buffer to overcome the frame variability. A Proportional–Integral (PI) controller, that is commonly used in control systems, is proposed to keep constant the occupancy of the buffer between the decoder and the display. A dead zone is defined to

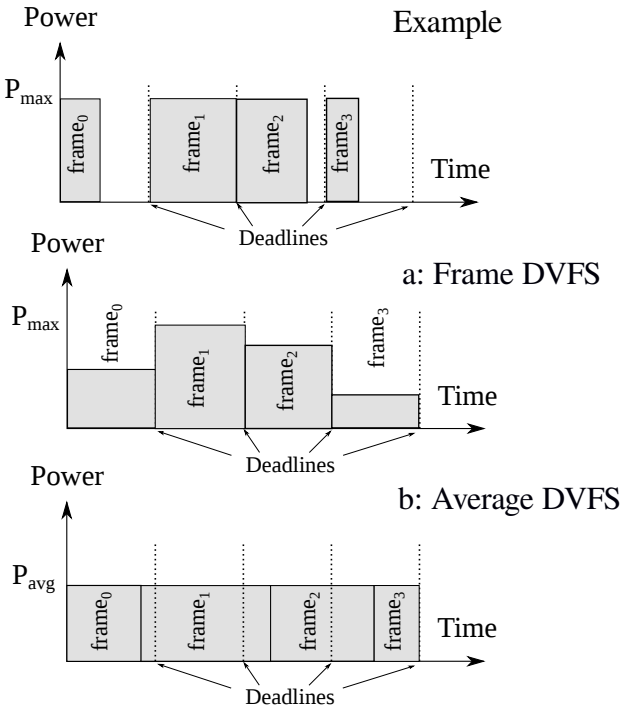


**Figure 5.2:** *Simple example with a) Frame by frame DVFS and b) average scaling policy*

prevent from deadline misses. In our approach, we also define a gauge where a critical level is defined to speed-up the systems if the occupancy is critically low.

Tan *et al.* [TMQW06] propose a prediction model that utilizes the block level statistics of each MPEG frame. The frames are subdivided into 9 sub-categories and a linear predictor is used to estimate the complexity of the sub-sequences. The coefficients are computed after a training on various sequences. In this chapter, the complexity is computed from the type of frames but a low pass filter is used to extract the first order statistics of the complexity. Combined with an outlier detector, the system is simpler to implement with no extra complexity that would consume energy. Moreover, no training on various sequences is required.

Similarly, Akyol *et al.* [AVdS08] propose to take into account the frame structure and to estimate online the decoding complexity with a prediction algorithm. Authors use a single core architecture model with an H.264/AVC decoder. The task is more challenging for HEVC on a multicore architecture since the decoding process varies more.

In a different fashion, Band *et al.* [BBYC09] use an adaptive filter to compute the complexity of decoding in a reactive way from the observation of the time needed to decode frames. Indeed, the authors consider that online estimation can be accurate. We propose to make the estimator more robust with an outlier detector.

To act proactively, Ma *et al.* [MHW11] have introduced a DVFS enabled video decoder using an a-priori knowledge on the video complexity. The complexity is computed by the encoder. The scheme requires the insertion of additional data, called Green Metadata, in the video stream. As a consequence, a modification of the encoder is needed to generate these metadata and a non trivial training phase in the decoder is necessary to utilize efficiently the metadata. This solution is proposed to be included as an option in MPEG standards [Gre13].

In the rest of the chapter, the average DVFS policy (based on Jensen's inequality) is called the Optimal DVFS (ODVFS) scheme. While being optimal in terms of energy consumption over a decoding period, Figure 5.2 clearly shows that deadlines are likely to be missed for individual frames in the sequence. It reduces then the QoS of the application if no countermeasure is taken to resynchronize the display.

### 5.2.5 Contributions of the Chapter

Firstly, most of the work [AVdS07, MHW11, CAMJ14] presented above do not consider the decoder as a complete system. Consequently, features like power consumption, latency and DMR are not analyzed jointly. By considering the complete system, this chapter analyzes the QoS of the system as a whole.

Moreover, studies using DVFS for video decoding [AVdS07, MHW11] are targeting previous video standards and not HEVC. Yet HEVC is a more challenging application. The modeling of the complexity that is used may be inefficient when targeting HEVC. By estimating the complexity on-line, the approaches proposed in this chapter use the actual decoding complexity of the on-going decoding task. Consequently, they are robust to any erroneous estimation of the decoding complexity.

For low power considerations, specialized circuits are usually proposed as described in Chapter 3 but they can not be deployed on the market for existing devices. By choosing to analyze GPP MPSoC, this chapter aims at providing solutions for HEVC to be implemented on former devices. Actual power performance are provided. They exhibit that software based decoders can be deployed realistically on existing devices when they are designed in a low power fashion with an appropriate low-power set-up.

This chapter proposes two methods that can outperform the state-of-the-art linux governors in finding low power execution scheduling. The first one uses thoroughly the big.LITTLE platform capabilities in terms of frequency scaling. It uses feedback information tracking the decoding complexity and can be considered as a reactive scheme. The second method uses the HEVC properties and can be considered as a pro-active DVFS scheme. We also show that these methods can achieve *close-to-optimal* performance with a limited implementation cost and no metadata. Finally, by measuring the performance on a real SoC, we show that HEVC can be performed efficiently with pure software design. Such a software implementation fosters a fast deployment of the HEVC standard and shows the high potential short-term market impact of HEVC.

## 5.3 HEVC Decoding System Analysis

The proposed energy efficient HEVC decoder is based on the open source *OpenHEVC* project [Ope] implemented on a low-power embedded platform based on a heterogeneous computing architecture of type ARM big.LITTLE.

### 5.3.1 Decoder Architecture and Performance

#### Architecture

The *OpenHEVC* decoder is developed in C programming language on top of the *FFmpeg* library [FFm]. The *OpenHEVC* decoder implements a conforming HEVC decoder and supports the three main profiles defined in the HEVC standard, namely Main, Main 10 and Main Still Picture profiles [SOHW12a]. As described previously in Section 2.3.5, HEVC decoding is readily processed in a parallel fashion. The principle of the frame level

parallelism as it is exploited in *OpenHEVC* is recalled in Figure 5.3. All decoding steps are performed at the level of a Coding Tree Block (CTB).

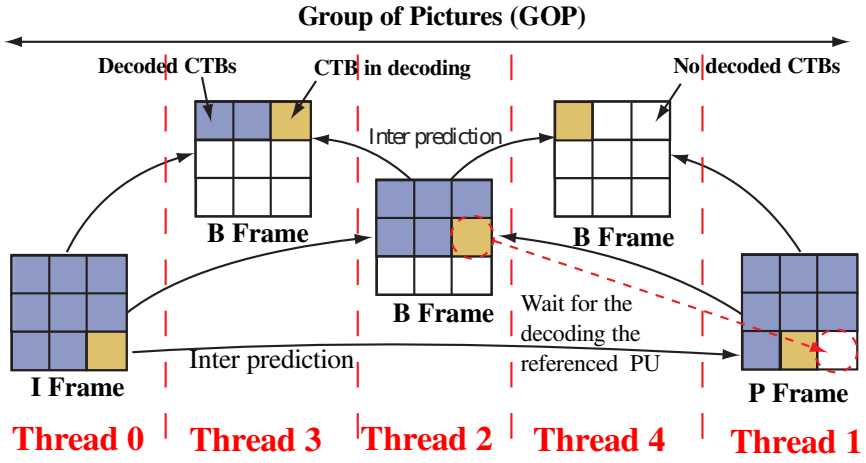


**Figure 5.3:** *Principle of the frame based parallelism in HEVC [HRD14a]*

It illustrates the concept of the frame based parallelism where all frames within a GOP are decoded in parallel. The frame-based parallelism requires communication between threads decoding different frames in order to ensure that the PU (Prediction Unit) used as reference for inter prediction is already available (decoded) in the reference frame. Therefore, the encoding/decoding of $n$ frames can be carried out in parallel.

When the frame-based parallelism is enabled in *OpenHEVC*, $n$ instances of the decoder are created, with $m$ the number of decoding threads. Each decoder has its own local and global structures, while some information in the global structure are shared between the $m$ decoders, such as the DPB and the lists related to the video headers. An inter-thread control solution is implemented to ensure that the PU required for motion compensation is decoded at the reference frame. Otherwise, the decoding thread waits until the PU is decoded in the reference frame. Once the required PU is decoded, all waiting threads are unblocked to pursue the decoding process.

### Profiles and Resolutions

To cope with numerous types of scenario, the MPEG standardization group defines several encoding profiles with different characteristics in terms of compression performance and coding complexity. The two main profiles are the RA profile and the Imain profile [SBS14].

The Imain profile mainly relies on spatial prediction. The main advantages of the Imain profile are the simplicity of encoding and the minimal transit delay. The RA profile is based on both spatial and temporal prediction. Inter-picture similarities are then leveraged, resulting into a high efficiency compression. The processing needed to encode with RA profile is higher than with Imain profile.

The complexity of decoding a video is closely related to its resolution (width x height) and frame rate (in Hz or Frame per Second (FPS)). Considering a handheld device, a wide range of resolutions can be found to address a high variety of devices. Table 5.1 lists the resolutions and frame rates of the sequences used in this study. The bit rate for a given resolution can be reduced if QP is increased, at the cost of a reduced image quality. The test sequences are 10 seconds long. Four configurations are available per sequence with $QP = 22, 27, 32, 37$. In addition to the reference streams, a *Mixed* sequence is created. It

concatenates *BasketballDrill*, *PartyScene*, *Cactus* and *BQMall* in order to test the effects of fast modifications in a decoded scene.

| Class | Sequences | Resolution | Frame rate (Hz) |
|---|---|---|---|
| B | *Kimono (K)* | 1920x1080 | 24 |
| | *Cactus (C)* | | 50 |
| C | *BasketballDrill (BD)* | 832x480 | 50 |
| | *PartyScene (PS)* | | 50 |
| | *BQMall (BQM)* | | 60 |
| | *RaceHorses (RH)* | | 30 |
| | *Mixed (M)* | | 30 |
| E | *KirstenAndSara (K)* | 1280x720 | 60 |
| | *FourPeople (FP)* | | 60 |

**Table 5.1:** *Video sequences considered in the experiments*

In this chapter, class C and class E videos are analyzed in particular as theses classes are the most frequent ones in the handheld devices context [You12]. We also analyze the class B Kimono sequence to extend the study.

### Parallelism and Complexity Trends

Recent studies have outlined the benefits of parallelism on the power efficiency of HEVC decoding [CAMJ14, RNH+15]. Raffin *et al.* [RNH+15] showed an efficient software implementation of the *OpenHEVC* decoder onto a GPP SoC. This decoder uses data and task level parallelisms. On a quad-core architecture, the speed-up is shown to reach 3.8 for the Imain profile and 3.0 for the RA profile.

Figure 5.4 plots the decoding complexity (in cycles per frame) over time respectively for Imain and RA profiles. The *Mixed* sequence is used and is decoded by *OpenHEVC* on the target big.LITTLE platform at a fixed processing frequency. For the All-intra configuration (left) it can be seen that the complexity varies by a factor of roughly 2 between scenes and that, within a scene, the complexity is varying slowly. For the Random Acess configuration (right), the complexity varies more between frames, depending on the types of prediction used in the decoding frames.

Finally, Figure 5.5 shows a *per-frame-type* classification that shows the complexity of decoding depending on the type of frame. It shows that I-slice decoding requires significantly more processing than P-slice decoding and that P-slice decoding requires significantly more processing than B-slice decoding. Therefore, the complexity from one frame to another can be drastically reduced if the frames are sorted by type. Table 5.2 analyses the mean and the standard deviation of the number of cycles needed to decode the *Mixed* sequence of Table 5.1 to illustrate this characteristic.

The autocorrelation is a tool providing statistics that estimate the theoretical autocorrelation of the observed data [BJR11]. Fig. 5.6 analyses the autocorrelation of the frame decoding complexity after selecting of B frames. The sample autocorrelation has significant

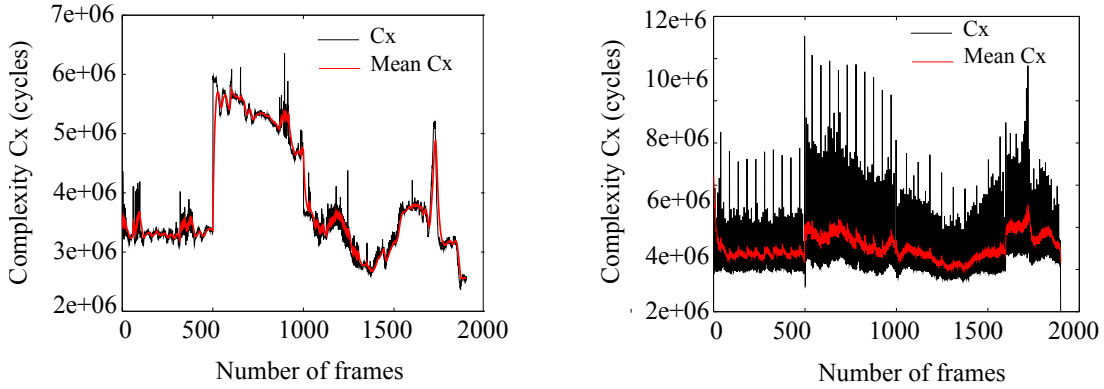


**Figure 5.4:** *Complexity and mean complexity from Mixed sequence for All intra profile (left) and Random Access profile (right)*

| | All | I | P | B |
|---|---|---|---|---|
| Mean | 4025 | 10572 | 5856 | 3329 |
| Deviation | 1791 | 571 | 634 | 791 |

**Table 5.2:** *Complexity statistical analysis (MCycles)*

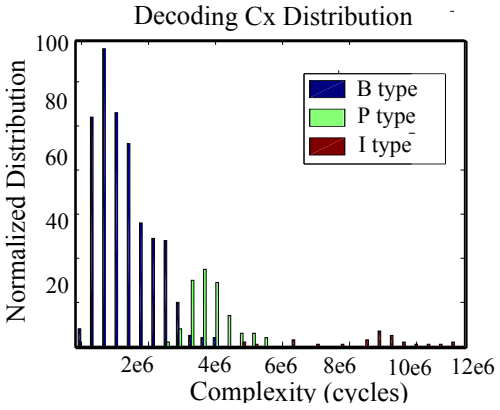


**Figure 5.5:** *Distribution of complexity in cycles for the different frame types*

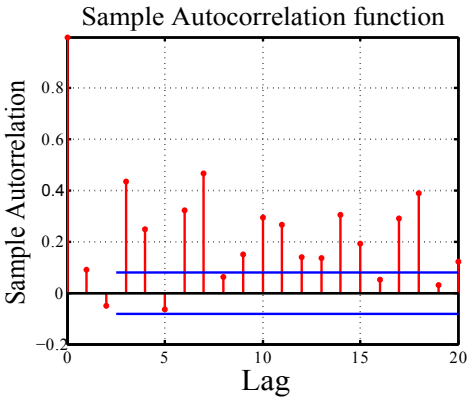


**Figure 5.6:** *Autocorrelation analysis the complexity decoding*

autocorrelation at different lag positions. It suggests that a Moving Average (MA) may be a suitable model to estimate the decoding complexity per frame type. A similar trend can be observed on I and P frames.

### 5.3.2 Performance Metrics

Figure 5.7 provides an overview of the HEVC decoding process at a system level.

The HEVC decoding process retrieves the input bitstream and generates uncompressed frames. The output frames are stored in a time-lag buffer before being delivered to the display unit. Large buffers prevent from deadline misses but add a latency to the system with more memory storage. For realtime systems, the *Decoding Rate* of the sequence shall be greater or equal to the *Display Rate*. It is typically expressed in *frames per second* (fps). Buffering takes care of deadline management but it leads to additional latency before video delivery. In use cases such as video telephony or live retransmission, this latency is strictly constrained because it lowers the quality of experience. Latency can also lead to memory

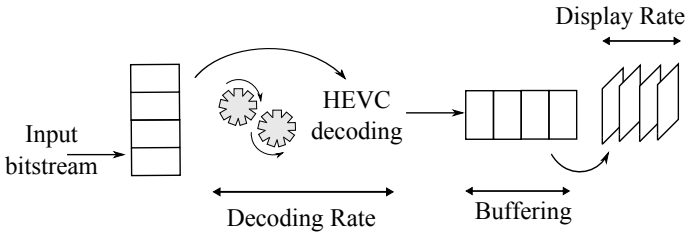


**Figure 5.7:** *Top level architecture of the video decoder*

issues due to the storage of uncompressed frames. For example, storing 20 frames of Full High Definition (FHD) video requires more than 40 Mbytes of memory which can be prohibitive for embedded systems.

## 5.4 Characteristics of the ARM big.LITTLE Targeted Platform

In this section, we detail the main characteristics of the target big.LITTLE system.

### 5.4.1 General set-up

The same platform as the one used in chapter 4 is considered in this study. The octa-core Exynos 5410 SoC is based on the big.LITTLE configuration including a cluster of 4 ARM Cortex-A15 cores and a cluster of 4 ARM Cortex-A7 cores. The SoC provides a set of 17 DVFS configurations from 250 MHz to 1.6 GHz. Only four cores can run at a time. From 250 MHz to 600 MHz, the SoC uses the Cortex-A7 cluster. From 800 MHz to 1.6 GHz, the Cortex-A15 cluster is used. Table 5.3 summarizes the set-up that is used throughout the study.

| | |
|---|---|
| SoC | Samsung Exynos5 Octa 5410<br>ARM Cortex-A15 Quad 1.6 GHz max.<br>ARM Cortex-A7 Quad 1.2 GHz max. |
| Memory | 2GB LPDDR3 @ 800MHz |
| OS | Ubuntu 14.04<br>with GCC 4.8.2 |
| HEVC decoder | OpenHEVC<br>branch shm6.1, version fc26f36... |

**Table 5.3:** *Experimental configuration*

### 5.4.2 Quasi-Heterogeneous Architecture

From an architectural point of view, the main characteristic is to support two types of processors with complementary processing capabilities and energy efficiencies. The little cluster hosts a quad-core A7 based GPP. The power efficient Cortex-A7 processor provides

an in-order 8-stage pipeline and exhibits 1.9 DMIPS/Mhz[1]. The big cluster hosts a quad-core A15 based GPP. The high-performance Cortex-A15 processor provides a multi-issue, out-of-order superscalar pipeline and exhibits 3.5 DMIPS/Mhz.

Even though it cannot be fully classified as a heterogeneous architecture, it has some of the corresponding characteristics. Indeed, the processing performance over a given function is different on the 2 types of cores. The main advantage of the architecture lies in the assembly code that can be executed by either cluster. Hence, the big.LITTLE architecture offers a wide range of scalabilities on processing functions.

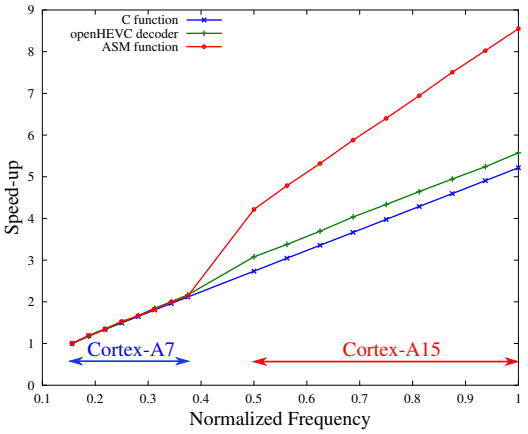


**Figure 5.8:** *Speed-up compared to the minimal frequency of the SoC depending on the application*

In Figure 5.8, the speed-up of processing compared to the execution with minimal frequency is depicted as a function of the processing frequency. The benchmark is performed on 3 types of functions: one coded in C, one in assembly and the *OpenHEVC* decoder that mixes both. As expected, if you consider the speed-up within a cluster, the speed-up is linear with the frequency. However, where one could expect a better speed-up slope on the Cortex A15 cluster, Figure 5.8 shows that it depends on the code efficiency. Pure C code offers a close-to-linear speedup when frequency increases. Assembly code can get a larger efficiency gain on the Cortex A15 cluster. For the *OpenHEVC* decoder implementation that mixes both C code and SIMD optimization in assembly [RNH+15], the speed-up is slightly better than with pure C code.

### Cluster Migration

The transition of processing functions from one cluster to another is performed by the cluster migration feature within the Linux kernel [Poi]. To make the system as transparent as possible, cluster migration is performed automatically according to frequency requested by the user. The transition was measured to take less than 20000 cycles [Sam12b] and is performed by the linux kernel. Rangan *et al.* [RWB09] states that moving a workload to an energy efficient core is more energy efficient than simply scaling down the processor frequency.

[1]DMIPS: Dhrystone Millions of Instructions per Second [Wei84].

### 5.4.3 Power Management

The big.LITTLE system comes also with power management provided by a frequency scaling management system. In Figure 5.9, the power dissipation per core is shown as a function of the frequency and number of active processors.

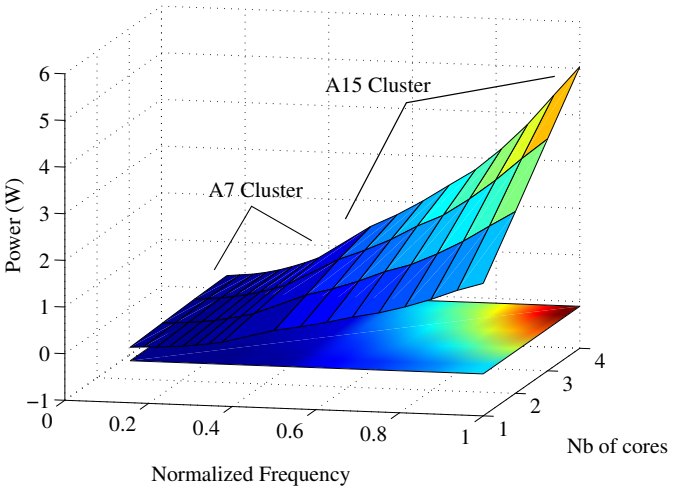


**Figure 5.9:** *Power trend for big.LITTLE systems*

Figure 5.9 shows that the big and LITTLE clusters have different power characteristics and the Cortex A7 cluster uses much less power than the Cortex A15 cluster. The frequency allocation manager must select as much as possible the A7 cluster and activate the A15 cluster only when strictly needed to respect the realtime constraints of the application.

The next section explains the contribution of combining advanced knowledge on the HEVC decoder and on the big.LITTLE hardware platform to optimize the energy consumption of the whole system.

## 5.5 DVFS Assisted HEVC Decoder

In this section, the low power HEVC decoder is presented. At first, the Optimal DVFS HEVC decoder that aims to minimize power consumption is presented. However, this proposal is shown not to be efficient for both Deadline Miss Rate (DMR) and latency. Therefore, two other alternative solutions are proposed utilizing both the HEVC decoder properties and the platform parameters.

### 5.5.1 State-of-the-Art Method 1: Optimal DVFS HEVC Decoder

The ODVFS HEVC Decoder minimizes dynamic power consumption using the power characteristics depicted in Figure 5.1. ODVFS considers the decoded sequence as a whole and perfectly aligns the decoding time to the sequence length. The ODVFS processing frequency $f_{opt}$ is given by Equation (5.3).

$$f_{opt} = C_{dec}.F_d \tag{5.3}$$

where $C_{dec}$ is the average decoding complexity in cycles per frames and $F_d$ is the display rate of the video. The average decoding complexity $C_{dec}$, in cycles per frame, is computed by Equation (5.4)

$$C_{dec} = \frac{1}{N_F} \sum_{i=1}^{N_F} C_i \tag{5.4}$$

where $C_i$ is the complexity in cycle of the decoding of a frame $i$ and $N_F$ is the total number of frames of the sequence.
$C_{dec}$ can be determined off-line by setting the processing frequency $f_{proc}$ arbitrarily. In practice, $f_{opt}$ is not always available in the DVFS frequency pool. The closest larger frequency $f_{ref}$ is then chosen, with $f_{ref} = min(f_{min}, f_1, ..., f_{max})$ s.t. $f_{ref} \geq f_{opt}$.

As explained in Section 5.2.3, the ODVFS method is likely to miss intermediate deadlines in the sequence. The next methods solve this limitation.

### 5.5.2 Proposed Method 2: Boosted DVFS HEVC Decoder

We propose the Boosted DVFS (BDVFS) HEVC decoder that aims to minimize the decoding latency by preventing any deadline miss throughout the decoded sequence. The BDVFS method consists in starting decoding at the maximum frequency available on the big.LITTLE SoC and then, once the output buffer is filled, adapting the frequency based on the decoder estimated complexity and the buffer filling level. This decoder falls into the category of reactive DVFS decoders, i.e. it reacts to the state of its consisting operations.

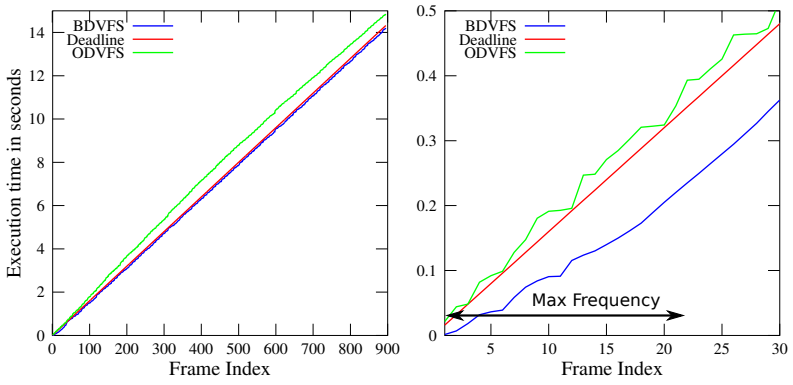


**Figure 5.10:** *Execution time of a 720p video decoding (60 fps) using Boosted DVFS and Optimal DVFS decoders. Left is the complete test and Right zooms in the heating-up zone.*

As depicted in Figure 5.10, the BDVFS first gets ahead of the deadline constraint while running at maximum frequency, thus avoiding the early deadline misses and delays of the ODVFS decoder. Then, it follows the realtime constraint, running as slow as possible but keeping a safety margin to prevent deadline misses due to the large variations of the decoding time from one frame to another, especially when the decoder is run on multiple threads. Figure 5.11 shows this phenomenon. Multi-threaded execution emphasizes the decoding time variation from frame to frame. Indeed, in the general case, perfect scaling onto the available cores speeds up the execution accordingly. However, this scaling can be inefficient in some cases. The difference between the minimum and maximum decoding time per frame is then bigger with multi-threaded configuration.

#### Architecture

In our early work [NBP+15], we proposed a video aware DVFS decoder based on feedback information from the display manager to minimize the slack time. We propose here to enhance this proposal by reducing the lag-time buffer.

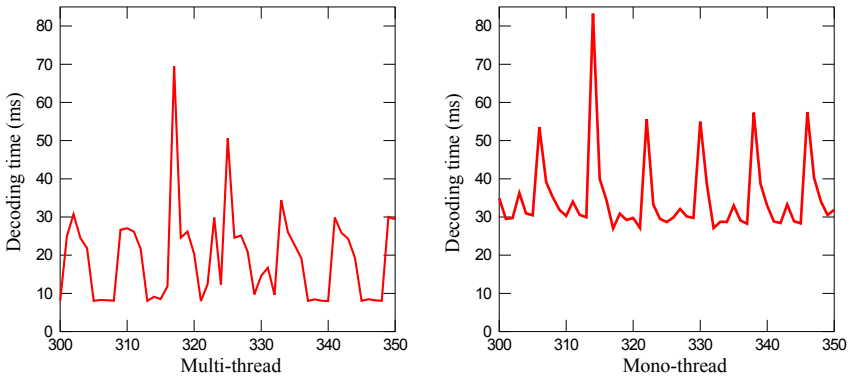


**Figure 5.11:** *Decoding time of a 720p video using mono-threaded and multi-threaded Optimal DVFS decoder*

The software architecture of the BDVFS HEVC decoder is shown in Figure 5.12. The *HEVC Decoder* and *Display Manager* functions are taken from the legacy *OpenHEVC* project [Ope].
The decoder takes an input bitstream and decodes each frame of the video and the display manager outputs the decoded frames at a fixed rate $F_d$ in fps set by the system.

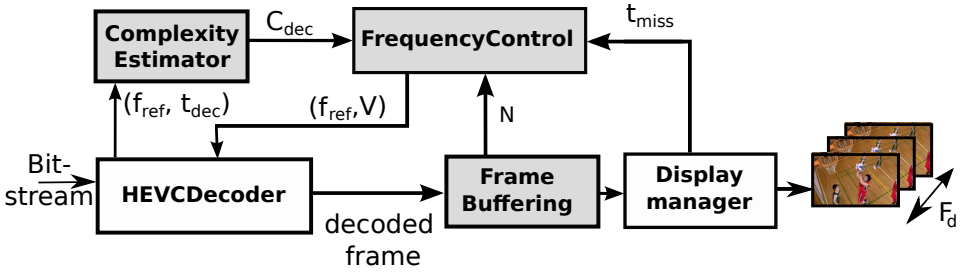


**Figure 5.12:** *Boosted DVFS HEVC Decoder software setup*

### Description

In order to find the minimum working frequency, and thus run as slow as possible, estimating the next frame decoding complexity $C_{dec}$ is essential. The complexity, expressed in clock cycles, can be computed from the decoding time of a frame, $t_{dec}$, and the processing frequency $f_{ref}$. To smooth out the complexity variations, a sliding window of size $L$ is used to track the average complexity. This function is described in algorithm 5.1.

Using the complexity $C_{dec}$ computed by the *Complexity Estimator* function, the *Frequency Control* function, described in algorithm 5.2, can then compute the minimum working frequency for the next frame to decode. In addition to the estimated complexity, this function takes into account the current buffer filling level and the potential miss time of the previous frame.
The optimal processing frequency $f_{ref}$ is first computed based on the actual deadline $T_{dl}$ and the complexity $C_{dec}$. Then, $f_{ref}$ is adapted if the buffer filling level exceeds the buffer high or low thresholds, respectively $T_h$ and $T_l$. $B_{shift}$ is proportional to the fullness of the lag-time buffer. It is used to speed-up (or slow-down) the processing frequency $f_{ref}$ by steps of $F_{step}$. $F_{step}$ is a tuning parameter set to the unitary DVFS step supported by the platform.

**Algorithm 5.1:** Estimation of a frame decoding complexity based on the average decoding time

**Input**: Current frequency $f_{ref}$;
Decoding time in seconds $t_{dec}$
**Param**: Sliding window length $L$ in frames
**Output**: Frame complexity $C_{dec}$ in cycles

**for** $i \leftarrow (L-1)$ **to** 1 **do**
  $C(i) \leftarrow C(i+1)$
**end**
$C(L) \leftarrow f_{ref} * t_{dec}$ // Update the window;
$C_{dec} = \frac{1}{L} * \sum_{i=1}^{L} C(i)$;

**Algorithm 5.2:** Frequency control process with missed deadline management

**Input**: Number of frames in the buffer $N$; Complexity in cycles $C_{dec}$;
Miss time in seconds $t_{miss}$
**Param**: $F_d$ frame rate playing demand in fps;
Available frequencies $[f_{min}, f_1, ..., f_{max}]$ in Hz;
Frame buffer size $N_{buf}$ in frames;
Frame buffer high threshold $T_h$;
Frame buffer low threshold $T_l$;
$F_{steps}$ Frequency number of steps;
**Output**: Frequency of the next period $f_{ref}$

$T_{dl} \leftarrow \frac{1}{F_d} - t_{miss}$ // Actual deadline computing;
**if** $T_{dl} < 0$ **then**
  $f_{opt} \leftarrow f_{max}$ // More than one frame behind;
**end**
**else**
  $f_{opt} \leftarrow C_{dec} * \frac{1}{T_{dl}}$ // Frequency scaling;
**end**
$f_{ref} \leftarrow min[f_{min}, ..., f_{max}]$ *s.t.* $f_{ref} \geq f_{opt}$;
$B_{shift} \leftarrow |N - \frac{N_{buf}}{2}|$;
**if** $N \geq T_h$ **then**
  $f_{ref} \leftarrow f_{ref} - B_{shift}F_{step}$ *s.t.* $f_{ref} \geq f_{min}$ // Buffer nearly full;
**end**
**if** $N \leq T_l$ **then**
  $f_{ref} \leftarrow f_{ref} + B_{shift}F_{step}$ *s.t.* $f_{ref} \leq f_{max}$ // Buffer nearly empty;
**end**

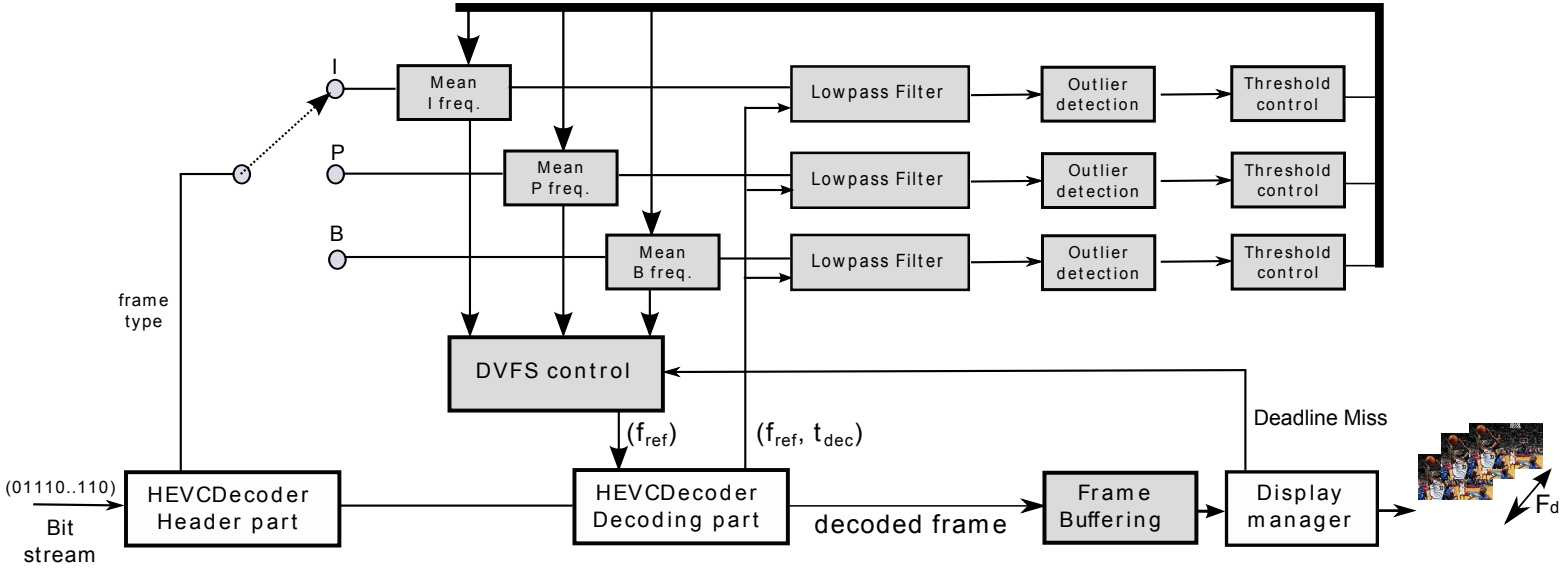


**Figure 5.13:** *Adaptive DVFS HEVC Decoder software setup*

### Characteristics

The Boosted DVFS HEVC decoder aims at keeping the buffer half full throughout the entire sequence decoding. Thus, the frequency shift between two steps is scaled based on the difference between the current buffer filling level and half of the buffer capacity. As depicted in Figure 5.14, this method insures a good frequency adaptation reactivity when the buffer is nearly empty (potential deadline miss) or nearly full (possibility to run slower).

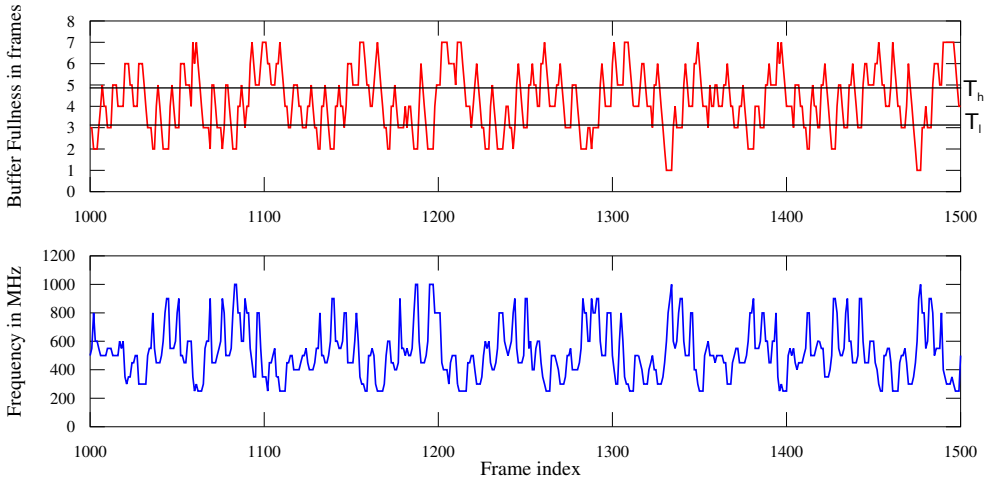


**Figure 5.14:** *Buffer filling level and working frequency for a 480p video decoding at 50 fps using BDVFS HEVC decoder*

Boosted DVFS obtains good results in terms of deadline compliance. However, its reactive form makes Boosted DVFS memory greedy because a large output buffer needs to be allocated. Moreover, this large buffering induces a long latency in the decoding. Adaptive DVFS, introduced in the next section, reduces this memory footprint and the decoding latency.

## 5.5.3 Proposed Method 3: Adaptive DVFS HEVC Decoder

The Adaptive DVFS HEVC Decoder aims at using the HEVC decoder properties to match the application with the underlying platform. Contrary to BDVFS which scales the frequency reactively, the Adaptive DVFS decoder adapts its processing frequency in a *quasi* pro-active fashion. The idea is to take into account the type of slice representative of the frame to predict its complexity. This information is extracted from the header part within the incoming bitstream.

### Architecture

The conventional decoder is split into two parts, header decoding and data decoding, as depicted in Figure 5.13. Once the type of frame is known, the complexity is computed per frame type, reducing drastically the estimation error on decoding time. The DVFS control manager can then decide the appropriate processing frequency to minimize the slack time. Frame buffering block and display manager block are similar to the BDVFS decoder. Algorithm 5.3 presents the different steps.

### Complexity Prediction

Adaptive DVFS (ADVFS) predicts the complexity on current decoding element from the characteristics of the video sequence (depicted in Figure 5.4). ADVFS is based on the complexity average (moving average) for each slice-type. The algorithm declares the average computing unit, using a low pass filter as shown in Equation (5.5).

$$C_{ma}(k) = a * C_{dec} + (1 - a) * C_{ma}(k - 1) \tag{5.5}$$

with $a$ as forgetting factor. This parameter can be tuned to react quickly to scene change or to smooth more the decoding time variations. The complexity is computed after header decoding and is used for the current frame to be decoded.

An *outlier detection* is added to remove any abnormal value that corrupts the complexity prediction process.

The last block is the *threshold control*, $T_{mult}$ in the decoder architecture. It is in charge of changing the frequency control regarding to the previous threshold, the type of frame and the possible deadline miss from the previous decoded frame. To refrain from constant changes, it is handled with a hysteresis, $H_H = 0.65$ and $H_L = -0.1$. The result of this process is a mean frequency for each slice type which is stored in a *Mean I,B or P frequency buffer* as shown in Figure 5.13.

The computed frequencies are sent to the *DVFS control* block which takes the nearest available frequency in the list of available frequencies.

### ADVFS Characteristics

Thanks to the pro-active management of the decoding complexity, the ADVFS decoder chooses the most appropriate frequency respecting frame-level realtime requirements. In Figure 5.15, the lag-time buffer behavior is plotted. Any frame shall be decoded before the deadline. In other words, any curves shall be below the dealine one. Any deadline infringement is managed with the adaptation process of the processing frequency. This is the main advantage when compared to the ODVFS scheme where deadline misses cannot be controlled.

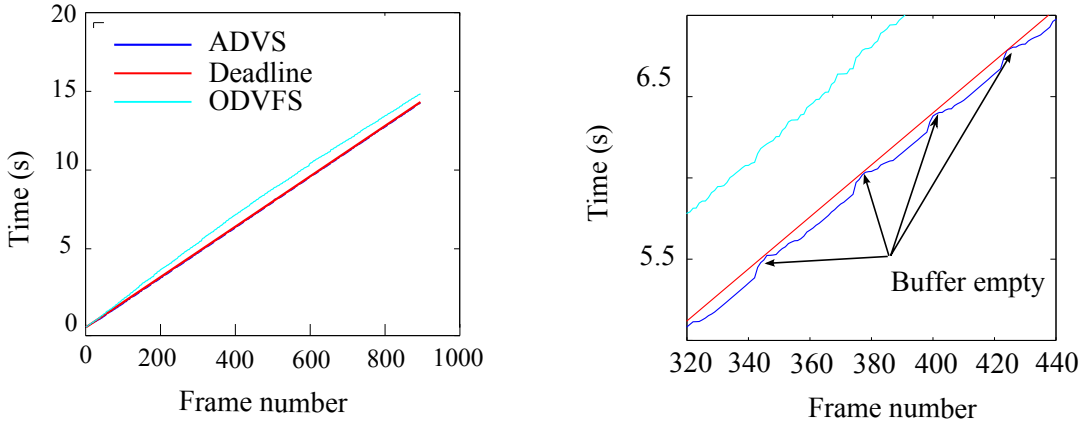


**Figure 5.15:** *Execution time of a 720p video decoding (60 fps) using Adaptive DVFS and Optimal DVFS decoders. Left side is the complete sequence and the right side zooms in to illustrate the buffer management.*

**Algorithm 5.3:** Frequency control per slice type

**Input**: Complexity in cycles $C_{dec}$
$Type$ type of prediction (B, P or I)
$F_d$ frame rate playing demand in fps
$DeadlineMiss$ indicator for display manager
**Param**: $T_{mult}$ Coefficient to scale the new frequency
$C_{ma}$ the running average complexity of the decoded sequence
Available frequencies $[f_{min}, ..., f_{max}]$ in Hz
Low and High threshold $T_l$, $T_h$
**Output**: Frequency of the next period $f_{ref}$

```
switch B, P or I Type do
    C_ma ← a * C_dec + (1 − a) * C_ma; // Calc. mean
endsw
if DeadlineMiss then
    if T_mult < T_h then
        T_mult ← T_mult + H_H;
    end
    else
        T_mult ← T_h;
    end
end
else
    if T_mult > T_l then
        T_mult ← T_mult + H_L;
    end
    else
        T_mult ← T_l;
    end
end
f_opt ← T_mult * C_ma / F_d; // Frequency scaling
f_ref ← min[f_min, ..., f_max]  s.t.  f_ref ≥ f_opt;
```

Besides the adaptive process reaches the close to optimal processing frequency. Indeed, by tracking the actual speed and comparing to the playback rate, the decoder performance trend reaches the optimal decoding speed after a few tens of frames. Figure 5.16 depicts how the efficient decoding rate is reached after 30 frames of decoding.

## 5.6 Performance Results

In this section the different versions of low power HEVC decoders are compared. The State-of-Art Linux governors are shown as references and the optimal energy decoder ODVFS (method 1) is compared to the two propositions (methods 2 and 3). Power performance and realtime performance are compared to get a full picture of the system achievement on the big.LITTLE SoC.

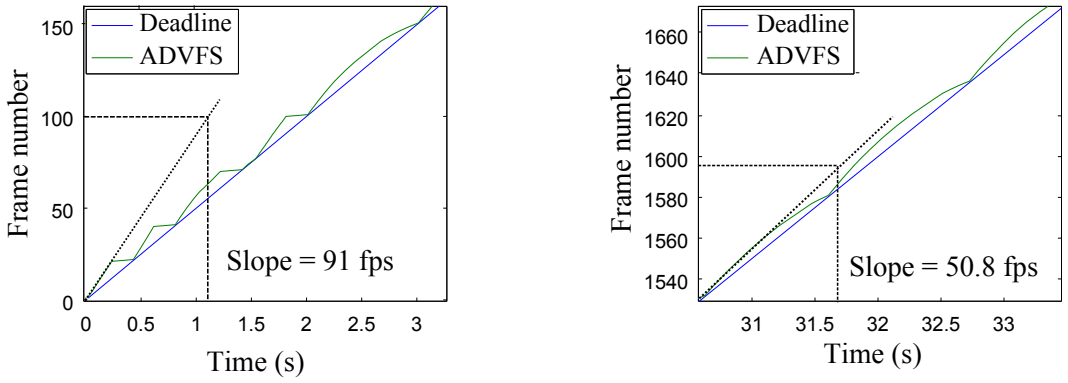


**Figure 5.16:** *Decoding speed adaptation of the ADVFS proposal - Targeted frame rate 50 fps*

### 5.6.1 Power Performance

The table 5.4 depicts a performance comparison of the power consumption. Different strategies are evaluated and the gain are computed relatively to the *Performance* governor as follows

$$G_x = \frac{100 \times (P_x - P_{perf})}{P_x}, \tag{5.6}$$

where $P_{perf}$ is the power consumption obtained with the *Performance* governor and $P_x$ the power consumption obtained with the considered strategy.

The results are given for different resolutions. The experimental set-up is described in table 5.3. Power measurements are made at a sample rate of 10 Hz on the SoC core. Both A7 and A15 clusters, as well as the memory consumption, are taken into account.

| | | 832x480 | | | | | | | | 1280x720 | | 1920x1080 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Video | | M | | RH | | BD | | BQM | | K&S | | K | |
| FPS | | 30 | | 30 | | 50 | | 60 | | 60 | | 24 | |
| QP | | qp27 | qp37 | qp27 | qp37 | qp27 | qp37 | qp27 | qp37 | qp27 | qp37 | qp27 | qp37 |
| Performance | Power | 1.19 | 0.96 | 1.41 | 1.02 | 1.65 | 1.27 | 1.88 | 1.52 | 2.76 | 2.41 | 3.51 | 2.86 |
| | Gain $G_x$ | - | - | - | - | - | - | - | - | - | - | - | - |
| OnDemand | Power | 0.49 | 0.32 | 0.66 | 0.37 | 0.89 | 0.59 | 1.18 | 0.76 | 1.73 | 1.28 | 3.15 | 2.01 |
| | Gain $G_x$ | **58.8** | **66.6** | **53.2** | **63.7** | **46.1** | **53.5** | **37.2** | **50** | **37.3** | **46.9** | **10.3** | **29.7** |
| Optimal DVFS | Power | 0.29 | 0.25 | 0.33 | 0.26 | 0.44 | 0.30 | 0.59 | 0.36 | 1.39 | 0.73 | 2.15 | 1.22 |
| | Gain $G_x$ | **75.6** | **73.9** | **76.6** | **74.5** | **73.3** | **76.4** | **68.6** | **76.3** | **49.6** | **69.7** | **38.7** | **57.3** |
| Boosted DVFS | Power | 0.31 | 0.26 | 0.37 | 0.27 | 0.54 | 0.31 | 0.81 | 0.4 | 1.39 | 0.89 | 2.2 | 1.23 |
| | Gain $G_x$ | **73.9** | **72.9** | **73.7** | **73.5** | **67.2** | **75.6** | **56.9** | **73.7** | **49.6** | **63.1** | **37.3** | **57.0** |
| Adaptive DVFS | Power | 0.3 | 0.26 | 0.35 | 0.28 | 0.45 | 0.31 | 0.76 | 0.38 | 1.4 | 0.87 | 2.22 | 1.22 |
| | Gain $G_x$ | **74.7** | **72.9** | **75.1** | **72.5** | **72.7** | **75.6** | **59.6** | **75.0** | **49.2** | **63.9** | **36.7** | **57.3** |

**Table 5.4:** *Power consumption (W) and gain (%) comparison for the different tested governors and DVFS methods*

The Linux *Performance* governor is taken as a reference and uses a race-to-idle stategy, i.e. it executes as fast as possible and puts the processor in a sleep state once a frame is successfully decoded. The *OnDemand* Linux governor uses DVFS. As expected, the *OnDemand* Linux governor performs better than the *Performance* Linux governor as it reduces dynamic power consumption. However, significant gains can be obtained by using video aware DVFS as in ODVFS, BDVFS and ADVFS. Gains of up to 76.6% can be achieved. Our proposed schemes, BDVFS and ADVFS have similar performance and very close to the optimal ODVFS. It can be noted that the gains compared to the *Performance* governor are higher with smaller slack time, i.e. when the realtime constraint is more stringent. It indicates a better management of DVFS by the proposed methods. It is particularly the case for high resolutions (1920x1080) or small QP (QP27) traducing a high quality video requiring more computing resources.

### 5.6.2 Optimal DVFS accuracy

One can argue that ODVFS cannot be perfectly implemented because of the discrete number of processing frequencies. Table 5.5 shows first the selected frequency to decode the sequence just-in-time. The selected input sequences lasts 60 seconds with a QP set to 27 on the DUT described in Table 5.3.

| Sequence | K | KS | BQM | BD | RH | Mixed |
|---|---|---|---|---|---|---|
| $f_{opt}$ (MHz) | 1000 | 800 | 600 | 500 | 400 | 350 |

**Table 5.5:** *Selected processing frequency to decode QP27 sequences*

Table 5.6 measures the relative error of the implemented ODVFS to a perfect just-in-time decoding, here 60 seconds. It shows that the error is negligible and confirms that the ODVFS can be considered as close-to-optimal from the power consumption point of view.

| Sequence | K | KS | BQM | BD | RH | Mixed |
|---|---|---|---|---|---|---|
| Time to decode (s) | 59.952 | 59.916 | 59.796 | 59.916 | 59.982 | 59.37 |
| Error (%) | 0.08 | 0.14 | 0.34 | 0.14 | 0.03 | 1.05 |

**Table 5.6:** *Relative error of the implemented ODVFS*

### 5.6.3 Selected Frequencies

A power efficient design can only be obtained by an efficient use of the available frequencies. It is especially true on a big.LITTLE system, where many frequencies are available. Figure 5.17 shows how the processing frequency is allocated as a video is being decoded. It shows that BDVFS makes an advanced use of the big.LITTLE architecture compared to the *OnDemand* governor. Indeed, BDVFS uses the energy efficient LITTLE cluster most of the time. The big cluster is used only from time to time for deadline alignment.

### 5.6.4 On-core - Off-core Power Consumption

The considered DUT is made from a SoC composed of on-core processing elements and off-core elements as given in table 5.3. The processing cores are embedded into the system and can use DVFS whereas the off-core elements like memory cannot be dynamically controlled. Because the proposed DVFS systems use a negligible memory the balance between on-core

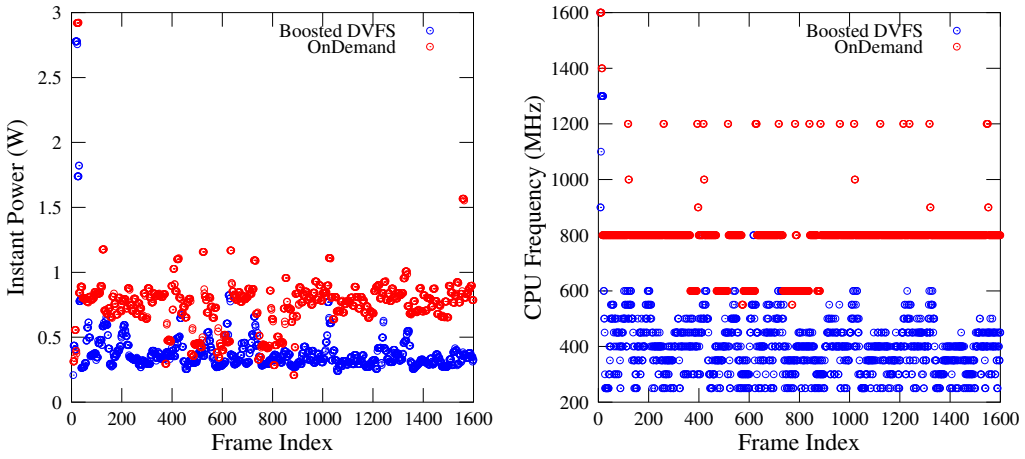


**Figure 5.17:** *Frequency utilization and instant power comparison of OnDemand governor and BDVFS for a sequence at 30 fps*

and off-core memory is analyzed for different video resolutions. The results are proposed in Figure 5.18.

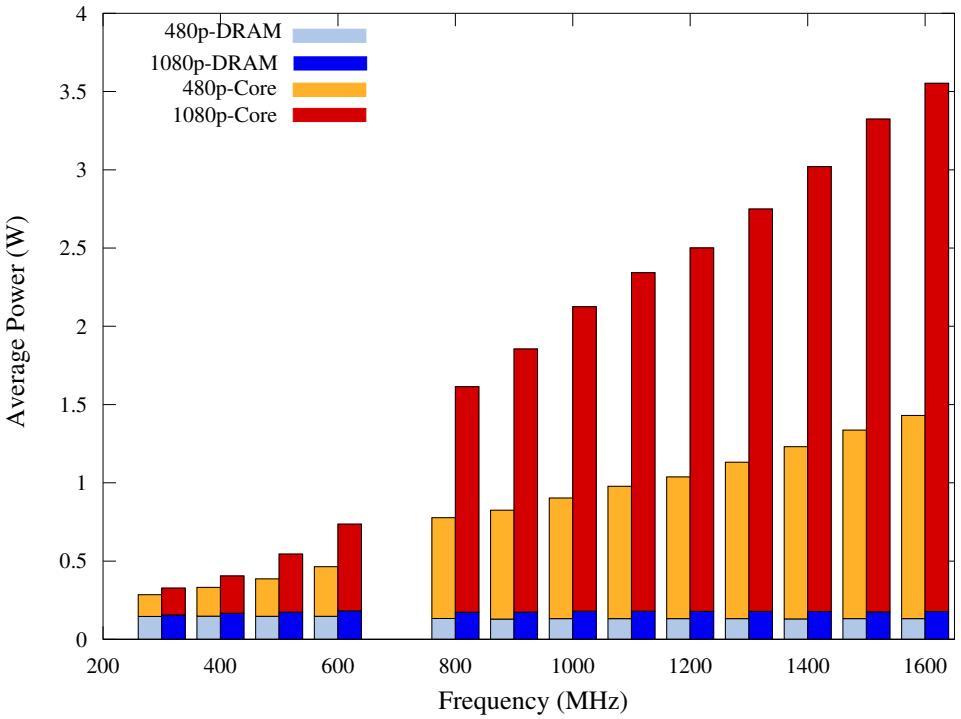


**Figure 5.18:** *Power consumption balance between on-core and off-core elements*

It notably shows that the memory impact is dramatically different from the processing resources that are needed. The power consumption of the memory is stable whatever the processing frequency. Therefore its impact on the complete system depends on the power needed by the processing part (*Core* in Figure 5.18). For not intensive decoding, the memory consumption represents half the power budget. For highly intensive decoding, the memory consumption is much less important and has a negligible role. For system designs, the memory impact shall be then taken into account together with the targeted use case.

### 5.6.5 Real Time and Latency performance

Section 5.6.1 demonstrates that low power design can be achieved by video aware DVFS decoders. Besides, while minimizing power consumption, the system shall fulfill the quality requirements. In other words, the video shall be delivered on time and overall latency shall be kept as minimal as possible. Section 5.6.1 recognizes the ODVFS as the best scheme as it uses the convex property of the power characteristics. The performance is computed on 1-minute long sequences.

| | Buffer | 480p | | | | 720p | 1080p |
|---|---|---|---|---|---|---|---|
| Video | | M | RH | BD | BQM | K&S | K |
| FPS | | 30 | 30 | 50 | 60 | 60 | 24 |
| Performance | 8-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| Ondemand | 8-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4-slot | 0 | 0 | 0 | 0 | $< 0{,}01$ | 0,084 |
| Optimal DVFS | 8-slot | $> 80$ | $> 80$ | $> 80$ | $> 80$ | $> 80$ | $> 80$ |
| | 6-slot | $> 80$ | $> 80$ | $> 80$ | $> 80$ | $> 80$ | $> 80$ |
| | 4-slot | $> 90$ | $> 90$ | $> 90$ | $> 90$ | $> 90$ | $> 90$ |
| Boosted DVFS | 8-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6-slot | $< 0{,}01$ | 0 | 0,040 | 0,045 | 0 | 0 |
| | 4-slot | 0,245 | 0,134 | 1,904 | 0,673 | 0,768 | 1,869 |
| Adaptive DVFS | 8-slot | $< 0{,}01$ | 0 | $< 0{,}01$ | 0 | $< 0{,}01$ | 0,28 |
| | 6-slot | 0,033 | 0 | $< 0{,}01$ | 0 | 0,039 | 0,251 |
| | 4-slot | 0,334 | 0 | 0,301 | 0,078 | 0,651 | 1,074 |

**Table 5.7:** *Deadline Miss Ratio (%) comparison of the different tested governors and DVFS methods*

However this scheme cannot guarantee realtime requirements unless a large time-lag buffer is used. Table 5.7 analyzes the DMR performance for all the presented policies. It shows that for an 8-slot time-lag buffer, the ADVFS and BDVFS can fulfill the realtime requirements. With a small time-lag buffer, the ADVFS proposal performs better than the BDVFS especially when the sequence to decode is more complex. To minimize the power consumption, both methods aim at converging to the average decoding speed which matches the play back rate. Therefore, the induced power consumption is similar. Any underestimation of the complexity leads to a risk of deadline miss which is monitored with the buffer fullness gauge. When the system requirements are more stringent (for example with HD video 4-slot buffer) then the ADVFS leads to slightly better results, the DMR drops from 1.8 % to 1 %.

## 5.7 Conclusion

In this chapter, we propose two novel DVFS-aware video decoders: BDVFS and ADVFS. They consider both the platform characteristics and the video decoding features. With on-line complexity estimation, they overcome the issue of load variability and respond to the challenge of implementing realtime decoding that was not possible with the model of the previous chapter. These decoders can be implemented into a video player targeting realtime play back.

From the performance point of view, they are compared to Linux governors and State-of-the-Art DVFS managements that are the typical options chosen by video players [LFCM07]. The BDVFS scheme performs reactive frequency changes while the ADVFS scheme performs pro-active frequency changes using HEVC decoding properties. The proposed schemes achieve power savings close to the upper bound that the ODVFS scheme can achieve and guarantee better realtime properties. We provide realtime analysis, displaying the DMR performance as a function of the length of the time-lag output buffer. These lag-time buffers are kept small with the proposals (from 6 to 8 frames) where previous studies on HEVC previously reported lag-time buffer of 50 frames [CAMJ14].

The proposed set-ups presented in this chapter envision new perspectives for the deployment of new standards like HEVC. This type of application is traditionally hardwired. But, we demonstrate that it can be fully implemented in software on modern SoC. And with a smart management of the resources combined with mechanisms like DVFS, decoders are not only realtime but also low power. A HD video can be decoded with a budget close to the watt. It means that *on-the-field* devices can be upgraded with HEVC with no hardware change and a limited impact on their power budget. User can then benefit from all the HEVC advances in terms of rate/distortion performance at the cost of a software upgrade.

# CHAPTER 6

## Modified Decoder for Energy Efficient HEVC Decoding

### 6.1 Introduction

Chapter 4 investigates low power design at the very early stages of the design. It proposes to find the best match between application QoS requirements and platform characteristics. No particular knowledge on the algorithmic content of the application is needed with this approach.

In Chapter 5, the focus is shifted to video applications. The chapter takes into consideration their characteristics at run-time and uses the platform features like DVFS to reduce power consumption.

In this chapter, we narrow the scope of application and focus on HEVC itself. Once all the techniques of the previous chapters are used, then the next step to achieve more power reduction is to analyze the application at an algorithmic-level. Indeed, applications like multimedia management are inherently error-tolerant and the computational accuracy can be played with so as to reduce the energy consumed by signal processing. This approach falls into the approximate computing research field. We propose to define and apply a framework of approximate computing at the application level to reduce the power consumption of HEVC decoders.

The rest of this chapter is organized as follows. Section 6.2 recalls the related work on approximate computing. A methodology that analyses approximate computing for signal processing applications at an algorithmic level is proposed in Section 6.3. The methodology is then applied to HEVC decoding in Section 6.4 and block transformation of the decoder is described in Section 6.5. Finally, experimental results are reported in Section 6.6 before discussing the possible applications and perspectives within the GreenMetada MPEG standard [FDM+15] in Section 6.7.

### 6.2 Approximate Computing

The approximate computing paradigm has been emerging for a decade and can produce significant improvements from circuit to system level. This chapter uses the property of image and video processing applications that consists in benefiting from the imperfect human perception to relax constraints on result accuracy. Therefore, approximate computing at an algorithmic-level can be considered to reduce the power consumption with quality

considerations at the application level. The proposed approximate computing methodology targets a limited portion of an application that lends itself to computation approximation.

As defined by Chippa *et al.* [CCRR13], errors due to approximate computing can be classified in two separate categories denominated *fail small* and *fail rare*. In order to guarantee a functional system, the impact of approximation errors on application QoS must be limited. In the *fail small* category, an error is either always present or very frequent and thus its amplitude must be limited. In the *fail rare* category, an error can have high amplitude and thus its probability of occurrence must be low for the impact on QoS to be admissible.

The maximum admissible error is depicted in Figure 6.1 versus its probability of occurrence. The techniques used for approximate computing must lead to errors located in the grey area. The shape of this grey area defines the QoS requirements of the application. Let's define a third approximate computing error category named *fail moderate*. This category corresponds to errors having both moderate probability of occurrence and moderate amplitude.

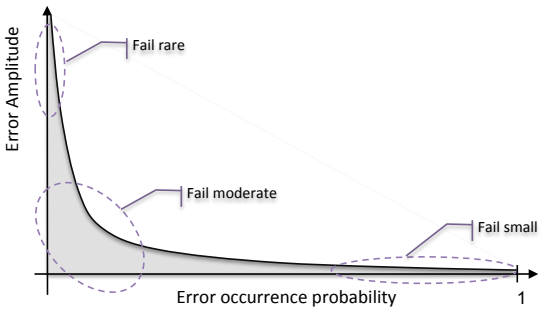


**Figure 6.1:** *Evolution of maximal error amplitude according to its occurrence probability*

At the technological level, voltage overscaling techniques can be used to reduce energy consumption. Working at near threshold regime or even at sub threshold regime achieves major energy consumption reductions but lead to computation errors due to an increase in circuit delay. The propagation time increases on the critical path leads to results unavailable at the clock edge and a mistaken value is considered. Nevertheless, these timing errors can be tolerated if the probability of occurrence is low enough (*fail rare*). Techniques have been proposed to compensate these errors [MCRR11] or to maintain these errors in reasonable bounds [KP11].

At a logical level, the functionality is approximated by simplifying logic. In [JS15], a Probabilistic Pruning technique is proposed to optimize the logic. The idea is to prune parts of the circuits which are rarely used and for which the approximation error due to pruning is moderate. Approximate computing circuits have been designed for specific functional units like adders and multipliers. In [LEN+11], Probabilistic Pruning is used to design an adder. Different adder circuits [DVM12, VC15, KK12] have been approximated with speculative techniques. The aim of this method is to relax the constraint on the critical path of the adder corresponding to the carry propagation chain. The carry is approximated with an intermediate carry chain using a limited number of stages.

In [CRRC11], the concept of Dynamic Effort Scaling is introduced. The aim is to adjust dynamically the tradeoff between energy and QoS. The QoS is evaluated with sensors and is used to regulate the level of approximation applied to the application.

Research work focus more on technological, logic and architecture levels. Algorithm level has not been studied in depth despite the significant energy gain opportunities. At

algorithm level, several techniques have been proposed and tested on different applications. In [LNC96], the concept of incremental refinement is introduced. The idea is to reduce the number of iterations of iterative processing. By skipping part of the processing, the energy can be reduced. In [CMR+10], the authors propose to explore the algorithm parameters in a Support Vector Machines algorithm to control the application complexity. The proposed techniques have been applied on particular applications. Nevertheless, to the best of our knowledge, no global approach has been proposed to explore algorithm level approximate computing.

Moreover, in these research works, approximate computing is considered at the level of a single signal processing block, tolerating either fail rare or fail small errors. In this chapter, we propose a novel method that consists at applying approximate computing at the level of an entire application. The different signal processing blocks composing an application are likely to tolerate different categories of errors. By classifying each signal processing block in an application and managing QoS requirements at algorithmic-level, the proposed method extends the gains of approximate computing.

In this chapter, we propose an algorithmic-level method to apply approximate computing to a complete application. The method decomposes the application into processing blocks which types define the classes of approximate computing techniques they may tolerate. By applying these approximation techniques to the most computationally intensive blocks, drastic energy reductions can be obtained at a limited cost in terms of QoS. In order to demonstrate the efficiency of the approach, this method is applied to a complete MPEG HEVC decoder to explore the tradeoffs between energy consumption and application quality. Energy reductions of up to 40% are obtained for a moderate degradation of application quality.

## 6.3 Proposed Method: Algorithmic-level Approximate Computing

### 6.3.1 Techniques of Approximate Computing

One of the main levers for reducing the energy consumption of a system is the reduction of its computational complexity. Indeed, when considering real-time systems, reducing the processing latency increases the slack time (i.e. the time of system off-time) and, thanks to DVFS and DPM energy management techniques, the frequency and voltage can be adjusted to decrease energy consumption [BD12][KSY+02]. For signal processing systems in the broad sense, energy consumption can be reduced by decreasing digital circuit activity due to arithmetic computation.

Approximate computing techniques at the level of a signal processing block can be divided into two classes as presented in Figure 6.2: the *processing-oriented class* groups together techniques altering computation while the *data-oriented class* groups together techniques modifying data characteristics. Figure 6.3 illustrates the different approximation methods. The next sections detail these two classes.

#### Processing-oriented Approximate Computing Techniques

Reducing application complexity can be achieved by skipping signal processing blocks or by providing new blocks of low complexity that approximate the computation result. The challenge of approximation techniques is to maintain the approximation error in bounds acceptable to the application QoS. Processing-oriented methods can be further divided into

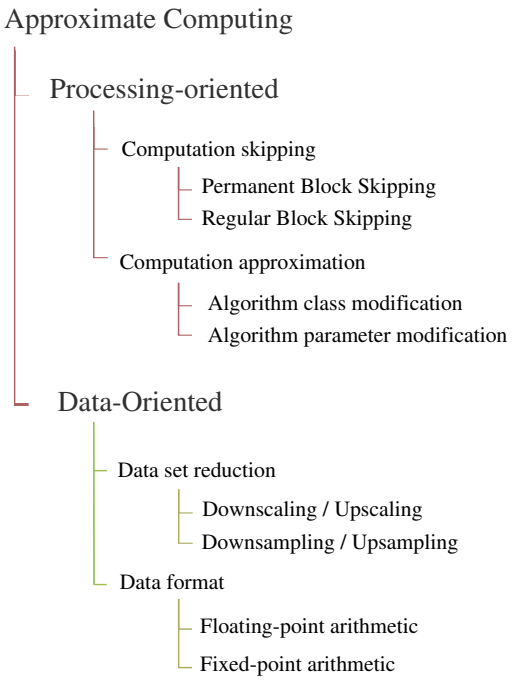


**Figure 6.2:** *Classes of approximate computing techniques*

computation skipping and computation approximation.

**Computation skipping.** This technique removes some parts of the processing in order to reduce computational complexity. In many systems, pre-processing and post-processing blocks aim at enhancing a signal. The processing associated with these blocks can be skipped permanently (*permanent block skipping*) or regularly (*regular block skipping*) without sacrificing drastically the application QoS. For example, in the audio compression domain, a technique called Spectral Band Replication (SBR) can be used to improve signal compression rate at high frequencies. The audio codec called MPEG High-Efficiency Advanced Audio Coding (HE-AAC) is obtained by adding SBR preprocessing to a standard Advanced Audio Coding (AAC) audio encoder. A computation skipping method can be applied to the SBR block to switch at encoder side between AAC and HE-AAC. HE-AAC can then be considered as the legacy application and AAC as the approximate low complexity application.

**Computation approximation.** This technique replaces a complex processing block by a different processing block, leading to a lower complexity. The two blocks should be mathematically equivalent. For instance, in some compilers, force reduction optimization is used to replace a complex operation by one of a lower complexity. As an example of force reduction optimization, the division by a constant is replaced by a multiplication with the inverse of the constant. More complexity reduction can be obtained by relaxing the mathematical equivalence between blocks. Computation approximation provides tools to explore the tradeoffs between the implementation cost and the errors introduced by the approximation.

A signal processing block can be approximated by preserving the same algorithm but using different parameters (*algorithm parameter modification*). For example, the order of a filter can be reduced to decrease its complexity.

Approximation can also be obtained by modifying further the algorithm in order to provide similar output but with less computation (*algorithm class modification*). For example, a Finite Impulse Response (FIR) filter can be replaced by an Infinite Impulse Response (IIR) filter. IIR filters requiring a lower order compared to FIR filter for equivalent frequency response characteristics, computation can be reduce while preserving original QoS.

Different techniques have been proposed to approximate elementary mathematical functions. Table-based approaches reduce drastically the evaluation time of a function but they require large memory capacities [dDT01]. Iterative approaches like CORDIC [And98] have been proposed for specific functions. In this case, evaluation time depends on the targeted accuracy. Polynomial approximation [CJL10] also provides a tradeoff between evaluation time and memory capacities.

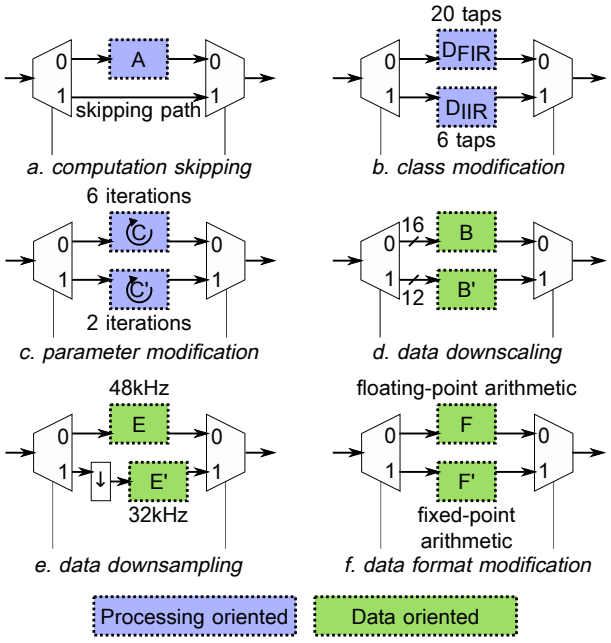


**Figure 6.3:** *Illustration of the classes of approximate computing techniques*

### Data-oriented Approximate Computing Techniques

Reducing the digital circuit activity at bit-level can be achieved by reducing the amount of data processed or by decreasing the number of bits used to code each data.

**Data set reduction.** This technique reduces the number of processed data samples in order to decrease the complexity. This method has an impact on processing because the complexity of an algorithm implementation is linked to the number of processed input data and the number of generated output data. Data set reduction can be applied in the time domain (*downsampling/upsampling*) by adjusting either the sampling frequency for signal processing applications or the frame-rate for video applications. As an example, music can be sampled at 32kHz or at 48kHz. The 32kHz processing represents an approximate, low complexity version of the 48kHz processing. The dimension of data can also be reduced by adjusting vector sizes for vector-based signal processing and image resolution for video

processing (*downscaling/upscaling*).

**Data format optimization.** Dynamic power consumption (i.e. power consumption related to the amount of processing) depends of the number of bit switches and thus is linked to the word-length of the data exchanged within the application. Thus, the power consumption depends on the arithmetic used to implement computation. For a given arithmetic, the aim of data format optimization is to reduce data word-length in order to decrease system activity. For instance, the same audio stream can be processed over 12-bit data or over 16-bit data. The 12-bit processing represents an approximate version of the 16-bit processing. Moreover, reducing data word-length decreases the latency of the application. Thus, energy gains may be obtained using DVFS and DPM energy management techniques when slack time appears. For fixed-point arithmetic, different techniques have been proposed for word-length optimization [PRMS10].

### 6.3.2 Classifying Application Signal Processing Blocks

The previous section detailed the different classes of approximate computing techniques that can be applied to a signal processing block. In order to determine which kind of technique to apply to each blocks composing an application, some decision criteria must be defined. A criterion can be defined according to the type of data generated by the signal processing block. Control-oriented and signal-oriented data are then distinguished.

**Blocks producing control-oriented data** affect the application control flow of the application. Thus, an error on a data can modify the control flow and change the applied processing. Such a modification alters in depth the application output. In this category, hard and soft control sub-categories are considered. In the **hard control case**, an error leads to corrupted data and no application output can be generated over a long period of time. In the **soft control case**, the modification of the control flow leads to an output error with high amplitude but with a limited effect in time. If the probability of occurrence of a soft control error is low enough, the degradation of the application QoS can be acceptable. This phenomenon corresponds to the fail rare case depicted in Figure 6.1. An example of fail rare errors are picture decoding errors in video codecs. Standalone Instantaneous Decoding Refresh (IDR) images, that can be decoded without any previous information, are regularly inserted in the video streams, typically with a period of 1 second. As a consequence, an error on image decoding cannot spread over more than a 1 second period.

**Blocks producing signal-oriented data** do not affect the application control flow and represent information with a determined precision compared to a reference value. A block that generates signal-oriented data can be of two types depending on whether the input and output data are in the same domain i.e. represent the same type of information (for instance time-based or frequency-based data). If the input and output data are not in the same domain, the signal processing block carries out a **domain transformation**. This kind of block can be approximated but not skipped. In the opposite case, the block is in charge of enhancing specific characteristics of the signal. This kind of **domain conservation** block can be skipped and/or approximated.

Figure 6.4 links the classes of processing blocks to the approximation techniques that may be applied to them. Efficient approximations are more likely to exist for signal-oriented

blocks. Thus, for a set of blocks with equivalent complexities, the block class should affect the order in which approximations are applied. In a first approximation step, domain conservation blocks should be considered because they are likely to support approximation better than other types of blocks.

Domain transformation blocks are considered only in a second step because a localized error in the input domain may spread in the output domain and affect much the QoS. Moreover, changing the data format from floating-point to fixed-point in a domain transformation block is likely to cause overflows and underflows in computation. In a third step, soft control-oriented blocks may be approximated. Special care must then be taken in the assessment of the QoS because the test vector must be representative of the soft control errors that will appear in the system. By definition, hard control-oriented block should not be approximated.

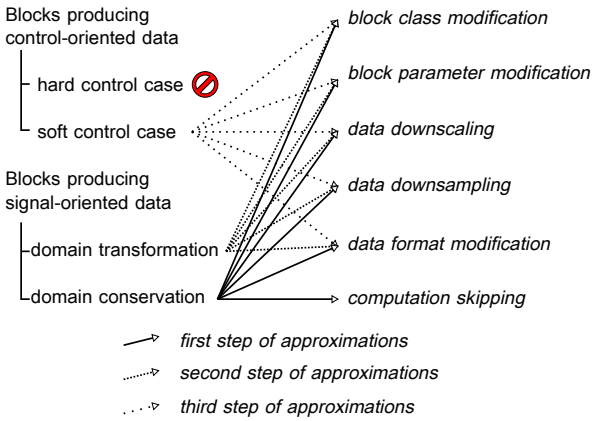


**Figure 6.4:** *From block types to approximate computing techniques*

### 6.3.3 A New Design Method for Algorithmic-level Approximate Computing

The new design method for algorithmic-level approximate computing is depicted in Figure 6.5. The application is modeled with a hierarchical block-based description. Dataflow diagram is a good candidate to describe the application. The first step selects the signal processing blocks that have the most important potential in terms of energy reduction by applying algorithmic-level approximate computing. The classification step, carried-out by the developer, identifies the class of each block in order to determine which approximate computing technique can be applied.

The profiling step reveals the complexity of each block relative to global complexity. Similarly to hardware or software optimization methods, blocks are to be processed in decreasing order of complexity. The output of this step is an ordered list of the application blocks and, for each block, the techniques that can be applied.

The second step checks for each signal processing block the potential benefits of approximate computing. Its objective is to explore the tradeoffs between energy reduction and QoS degradation. Firstly, the reduction of the implementation complexity is evaluated. The considered block is left unmodified if the potential gain appears to be negligible. In order to quickly evaluate the potential complexity reduction, a low complexity version of the block is compared to the legacy version. To obtain this comparison, the reference C code of the application is modified to implement the approximated version. The QoS

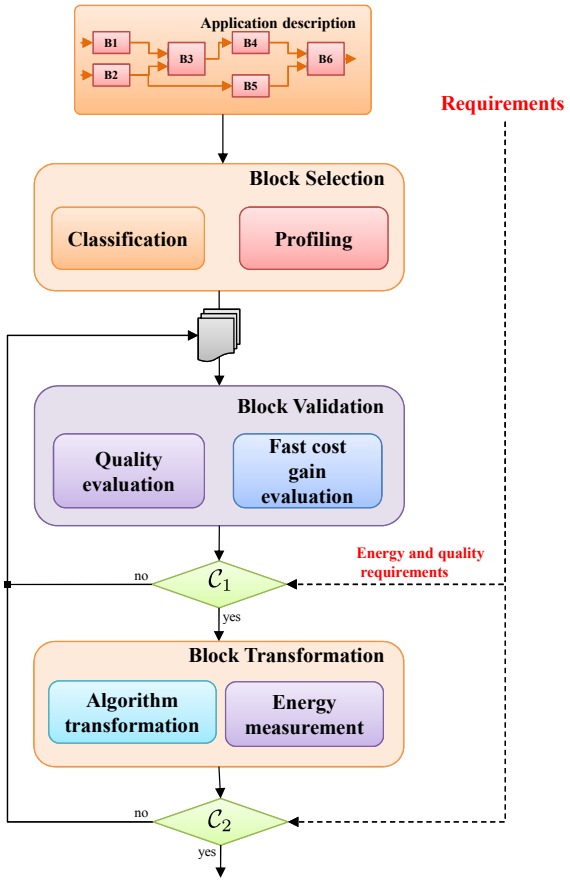


**Figure 6.5:** *Design flow for approximate computing at algorithm level*

degradation is evaluated from a set of simulations on a testbench. Complexity reduction is the criterion for the acceptance of the approximation. If the condition $\mathcal{C}_1$ on complexity reduction and QoS degradation is satisfactory, the third step corresponding to algorithm transformation is carried-out.

The aim of the third stage is to develop the optimized algorithm versions and to evaluate or measure the real energy consumption reduction. In the illustration example, code optimization at assembly level has been carried-out to exploit SIMD NEON vector processing. Step 2 and 3 are repeated for each block of the list generated with step 1 and for each algorithmic transformation if the QoS degradation is still acceptable (condition $\mathcal{C}_2$).

## 6.4 Algorithmic-level Approximate Computing of an HEVC Decoder

### 6.4.1 Decomposition of HEVC Decoding into Signal Processing Blocks

The standard structure of the HEVC decoder is depicted in Figure 2.2 [SOHW12b]. The overall description is given in the previous chapters.

The basics of the signal processing elements are recalled here. When receiving compressed data, the entropy decoder first extracts the different syntax elements from the video

stream using arithmetic coding, then the residual data are dequantized and transformed using an inverse DCT process, resulting in the error of the block intra or inter prediction.

Analysing the different types of scenario, the HEVC standard provides different profiles. RA configurations are used typically for broadcasting and use a pyramidal structure for picture reordering, i.e. different "levels" of prediction are defined and pictures of a level are predicted from pictures of lower levels only. The reference image, also called I-type frame, is sent periodically and all other frames are deduced from each other with the inter-frame prediction. Therefore two types of frames can be highlighted: the ones that are totally independent from the others and the ones deduced from the others. This property can be used to prevent any drift of behavior and is used later in this chapter. It shall be also noted that in AI configurations, all pictures use I-type of encoding and only intra-frame prediction is employed.

### 6.4.2 Classification of the Signal Processing Blocks

Based on the classification tree presented in Figure 6.2, the target application is analyzed at a coarse granularity. The HEVC decoder can be represented as depicted in Figure 6.6.

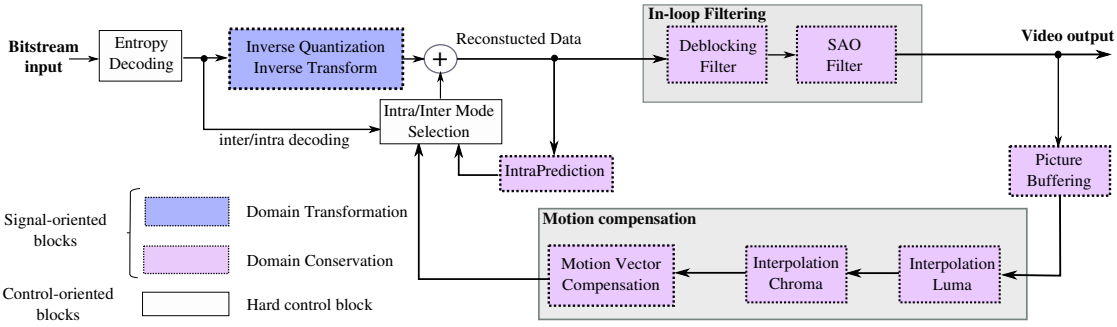


**Figure 6.6:** *Algorithm classification of an HEVC decoder*

The entropy decoding block is a hard control block, as each decoded value controls the type of processing executed on the data stream. Any approximation on its result would potentially break the downstream execution pipeline because wrong headers for a sequence, picture or block could be inferred. The inverse transform block performs a DCT transformation and thus is a (signal-oriented) domain transformation block. In-loop filters and motion compensation blocks are all signal oriented domain conservation blocks (filter). Intra prediction is also a signal oriented block consisting in replicating pixels in a given direction so as to predict a new block.

Intra/inter mode selection is a simple switch that is control-oriented and requires very few resources. Picture buffering corresponds to a queue of images retained for display and future picture predictions.

Note that this coarse grain analysis could be refined to look for further approximate computing opportunities, for example by separating inverse quantization and inverse transform.

### 6.4.3 Profiling Results

The HEVC decoding process has been profiled in [BBSF12, CPG+13, RNH+15, HRD14b] on various platforms such GPP, DSP and with different types of encoding such as RA and AI for different levels of compression and use cases. Our experiments confirm the trends

observed in these studies and are displayed in Figure 6.7. The experiment runs *OpenHEVC* decoder [Ope] on a Exynos SoC as described in Section 6.6.1.

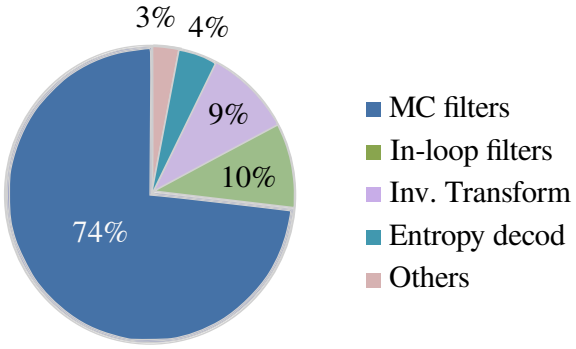


**Figure 6.7:** *Average Relative complexity of data processing blocks in an optimized decoder on a general purpose processor. Input sequences: Kimono RA 1920x1080 with Quantization Parameter from 22 to 32 by steps of 2*

More detailed results are provided for different quantization parameter values in Table 6.1. Results show that Motion Compensation (MC) filters require most of the computation resources. In-loop filters and the inverse transform require each about 10% of the processing time. Finally, entropy decoding is less computation intensive compared to other blocks. This fact can be explained by the amount of data processed by each block. Entropy decoding processes compressed data while filters, namely the MC filters and in-loop filters, process uncompressed data.

| Quant. param.(QP) | 22 | 24 | 26 | 28 | 30 | 32 | Avg |
|---|---|---|---|---|---|---|---|
| MC filters | 62.5 | 69.9 | 72.3 | 75.3 | 78.1 | 80.2 | 73.7 |
| Inv. Transform | 13.2 | 11.6 | 9.5 | 8.7 | 8.0 | 7.4 | 9.3 |
| In-loop filtering | 13.9 | 10.0 | 10.5 | 9.2 | 7.8 | 7.1 | 9.6 |
| Entropy decod. | 6.4 | 5.0 | 4.2 | 3.6 | 3.3 | 2.2 | 4.2 |
| Others | 4.0 | 3.5 | 3.5 | 3.2 | 2.8 | 3.1 | 3.2 |

**Table 6.1:** *Complexity analysis per processing blocks on an HEVC decoder*

Due to the high relative complexity of filtering functions, a substantial reduction of the overall decoding energy can be obtained by reducing the complexity of the filters. Their classification is illustrated in Figure 6.6 as domain conservation signal-oriented blocks respecting the classification of Figure 6.4.

### 6.4.4 HEVC Decoder Block Selection

Blocks are considered for approximation in decreasing order of computational complexity.

#### Motion Compensation Filters

In all recent MPEG codecs, a motion estimation technique is used by the video encoder to send a compressed information to a decoder by exploiting the temporal redundancy of the transmitted video sequence. A motion vector is defined for each block in the picture as the relative position of the predicted block with respect to the reference block in a previously decoded picture. However, the true movements of the blocks are not perfect translations

and can not perfectly match the sampling rate of the digitalized video. Therefore, a fractional accuracy is used for the motion vectors to reduce the prediction residual error and improve the compression performance. If a motion vector has a fractional value, the reference block needs to be interpolated accordingly to generate the prediction.

In the MPEG HEVC standard, the fractional motion vector compensation is performed by two separable 1-D interpolation filters for the horizontal and vertical directions [SOHW12b]. Representing 74% of the decoding effort and being signal-oriented domain conservation blocks, these filters could be skipped [NHP+14, NRPM15c] and should be tackled first in the algorithmic-level approximate computing method.

#### In-loop Filters

The in-loop filters are composed of two entities: the deblocking filter, already present in the previous H.264 standard and the SAO that has been newly adopted in HEVC. The in-loop filters have the particularity of processing decoded frames and can therefore be considered as enhancement functions to improve the decoding QoS.

The SAO filter is used to reduce sample distortion by classifying reconstructed samples into categories, obtaining an offset for each category, and then adding that offset to the sample value [FAA+12]. Each offset is calculated by the encoder and explicitly signaled in bitstream. SAO consists of two separate modes selectable by the encoder for each block called Coding Tree Units (CTU): Band Offset (BO) and Edge Offset (EO). When using BO, the classification of samples is done based on the sample values. When using EO, classification is done based on neighboring samples. The method to add the offset is similar in both modes.

The deblocking filter processes decoded frames to reduce the artifacts generated by inaccurate predictions. A blocking effect may appear between blocks because of the block transform for residual coding and it may be more visible near block boundaries. The deblocking filter enhances the decoded frames overall QoS by smoothing the transitions between blocks.

In-loop filters represent 10% of the decoding effort and they are signal-oriented domain conservation blocks, these filters need to be addressed second in the algorithmic-level approximate computing method.

The next block in terms of computational complexity is the inverse transform block. It is a domain transformation block, which approximation can spread over a large set of output data. As the 2 first candidates already represent 83.3% of the global decoding complexity, the next blocks including inverse transform block have not been considered for approximation.

### 6.4.5 Quality Evaluation for HEVC Decoder Block Validation

Defining metrics to evaluate the QoS of an application is a complex task. In this section, three usual quality evaluation tools for decoded videos are presented. HEVC compression is in general a lossy operation, i.e. it degrades the quality of the encoded video. This is acceptable because strict picture exactness is usually not necessary to the end-user of the video decoder. Thus, acceptable quality bounds for a decoded video need to be defined and the quality metrics shall be as close as possible to human perception.

### PSNR

In video compression domain, the Peak Signal To Noise Ratio (PSNR) metric is commonly used as a distortion metric [OSS+12a]. The metric is a combined PSNR of the luminance (Y) and the chrominance (U,V) components per image with different weights, $PSNR_{YUV}$.

$$PSNR_{YUV} = (6PSNR_Y + PSNR_U + PSNR_V)/8, \tag{6.1}$$

where $PSNR_Y$, $PSNR_U$ and $PSNR_V$ are independently computed as follows:

$$PSNR = 10log_{10}(d^2/e_{MSE}), \tag{6.2}$$

and where $d$ is set to 255 for 8-bit quantization and $e_{MSE}$ is the Mean Square Error of the reference image to the decoded image. The PSNR of the video is computed by averaging the PSNR per image.

PSNR is a widely used image distortion metric but is appears to be far from the human perception. For instance, shifting a whole picture of 1 pixel to the right may result in a small PSNR (signifying a bad quality) while the eye will detect almost no quality loss.

### Similarities quality evaluation

While PSNR is mainly based on a pixel-by-pixel approach evaluating the intensity differences of distorted and reference image pixels, other methods were developed to better match the perceived visual quality. Within this scope, Wang *et al.* [WBSS04] introduce an alternative framework for quality assessment based on the degradation of structural information called Stuctured Similiraty (SSIM). Assuming that the human eye is more sensitive to the image structural modifications, the SSIM metric measures the similarity between two non-negative image signals $x$ and $y$ of size $N$ which are supposed to have been aligned with each other.

$$SSIM(x,y) = \frac{(2\mu_x + C_1)(\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{6.3}$$

where $\mu_x$ is the mean of $x$, $\mu_y$ is the mean of $y$, $\sigma_x$ is the variance of $x$, $\sigma_y$ is the variance of $y$, $\sigma_{xy}$ is the covariance between $x$ and $y$ as defined in Equation (6.4). The terms $C_1 = (k_1 d)^2$, $C_2 = (k_2 d)^2$ are two variables to prevent unstable results when the denominator is small, $k_1$ is set to 0.01, $k_2$ is set to 0.03 and $d$ is the pixel range set to 255 for 8-bit quantization.

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \tag{6.4}$$

In the SSIM method, the quality is evaluated on luminance components only [WBSS04].

### Subjective Evaluation

Within the video decoding applications, the end-user is almost always a human observer and visual perception is the final arbiter of the visual experience. Therefore, having a large panel of end-users evaluating the application results is the most efficient way to evaluate the QoS. The metric is then called Quality of Experience (QoE) and its evaluation can be adapted to the actual services provided by the considered application.

To assess the video quality for multimedia applications, the Video Quality Experts Group (VQEG) gathers international experts that specify procedures of subjective evaluations [VQE08]. Since the purpose of the quality assessment is to evaluate whether the

errors induced by the approximate computing framework are acceptable by the end-user, the Double Stimulus Impairment Scale (DSIS) method Variant II, with a continuous impairment scale, can be selected to perform the subjective quality assessment experiment. This method was also chosen to analyze the quality performance of HEVC providing reference values [HRDSE12]. In a DSIS test, sequences are shown to the subject by pairs sequentially. She/he shall rate the quality of the second stimulus as decribed in Figure 6.8. The complete test sequence includes the reference video that is expected to have the best quality. The used rating scale is shown in Figure 6.8.

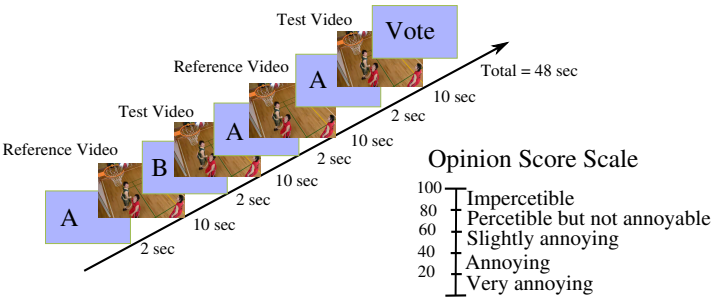


**Figure 6.8:** *DSIS Variant II and Opinion score scale*

A subjective video quality test session shall last less than 30 minutes to keep the concentration of the subjects high [p9192]. It is also recommended to select 30 subjects with a large range of ages. And subjects are supposed to be non-expert. All subjects shall be tested for acuity and color vision using Snellen charts and Ishiara charts.

Even if subjective testing is the most efficient tool to assess video quality, launching a test session is a heavy task. As an example, Hanhart *et al.* performed a 30-sequence test comparison on HEVC videos [HRDSE12]. The test session has last 2 days employing 34 non video expert subjects.

### Discussion on the Choice of a QoS Metric

The proposed design flow as described in figure 6.5 is based on an iterative process with a regular evaluation of the quality of the application. This quality evaluation is performed after each step of approximation of the original computation. Therefore, the QoS evaluation needs to be performed many times. Table 6.2 compares the time and cost of the PSNR, SSIM and DSIS evaluation tools.

| Method | Time | Cost |
|---|---|---|
| PSNR | 342 sec | - - |
| SSIM | 6810 sec | - |
| DSIS | 2 days | +++ |

**Table 6.2:** *Quality evaluation tools : Comparison for 30-sequence comparison test on a typical desktop computer*

For video applications, DSIS is the most precise quality assessment tool. However it cannot be used during the iterative part of the proposed design flow. SSIM is more appropriate as its outputs are closer to the human perception than the classical PSNR metric [HZ10]. It shall be noted that PSNR is more widely used especially within standardization bodies (e.g. MPEG). Hence it can be considered as a basic quality evaluation tool.

## 6.5 HEVC Decoder Block Transformations

### 6.5.1 Block Class Modification of the Motion Compensation Filters

As a conclusion of the discussion in Section 6.4, the first blocks to approximate are the MC filters as they represent the highest share of the processing load. Our first proposal is based on a reduction of the filtering complexity which is a block class modification (see Section 6.3.1). The modification is described in Figure 6.9 where the legacy filters are replaced by approximated version. An *approximation level control* signal chooses dynamically the filter complexity.

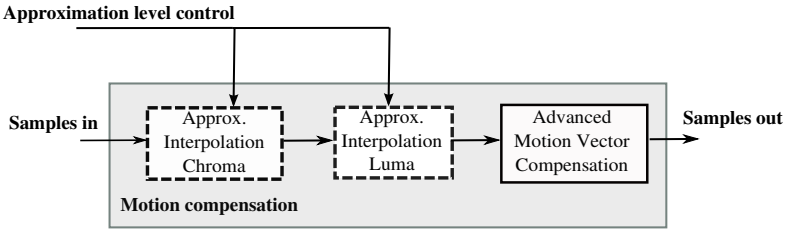


**Figure 6.9:** *Interpolation filter of the MC block approximation*

### 6.5.2 Complexity Reduction of MC Filters

The HEVC legacy interpolation filters are analyzed in details in [KAA+13]. They use coefficients generated from a Discrete Cosine Transform (DCT). Assuming a local list of pixels $\{p_i\}(i = -M+1, .., M)$ of $N_h = 2M$ samples, the forward DCT generates the Fourier coefficient $C_k$ (Eq. 6.5). The pair of forward-inverse transforms can be pre-calculated and merged for fractional position [KAA+13].

$$C_k = \frac{2}{N_h} \sum_{l=-M+1}^{M} p(l) \cos\left(\frac{(2\ l - 2 + N_h)\ k\ \pi}{2\ N_h}\right) \tag{6.5}$$

The HEVC standard uses FIR filters to perform the luminance and chrominance interpolation. Tseng *et al.* [TL08] have proposed the design methodology adopted by the MPEG HEVC standard. The resulting interpolation filter taps are generated from the DCT of Equation (6.5). They are named DCT-based interpolation filter (DCTIF) and the HEVC standard uses an odd or even number of coefficients depending on pixel position [TL08, LWX+12]. Even-length filters are computed with the following equation for any fractional position $\alpha$ (e.g. $\alpha = 1/2$ for half-pel coefficients).

$$h_m^e(\alpha) = \frac{1}{M} \cos\left(\pi \frac{m - \alpha}{2M - 1}\right) \sum_{k=0}^{2M-1} (c_k^2 . \phi_k) \tag{6.6}$$

with $\phi_k = \cos\left(\frac{(2m-1+2M)\pi k}{4M}\right) \cos\left(\frac{(2\alpha-1+2M)\pi k}{4M}\right)$.

Odd-length filters are obtained with the following equation for any fractional position $\alpha$.

$$h_m^o(\alpha) = \frac{2}{2M + 1} \cos\left(\pi \frac{m - \alpha}{2M}\right) \sum_{k=0}^{2M} (c_k^2 \phi_k) \tag{6.7}$$

with $\phi_k = \cos\left(\frac{(2m+1+2M)\pi k}{2(2M+1)}\right)\cos\left(\frac{(2\alpha+1+2M)\pi k}{2(2M+1)}\right)$.

Whereas the actual standard proposes a fixed configuration for the filters, we propose to generate a wide range of filters from the minimum size to the legacy size.

For example, in HEVC [SOHW12b], the 8-tap filter designed for the luminance is using $N_h = 2M = 8$. For the chrominance components, the standard filter size is fixed to $N_h = 4$.

We define five categories of filters relatively to their complexity: low, middle, intermediate, high and legacy respectively with $N_h = 1, 3, 5, 7, 8$ for luminance and with $N_h = 1, 2, 3, 4$ for chrominance.

Table 6.3 and Table 6.4 describe examples of original and modified filters for interpolation factors $\alpha$ standardized in HEVC. The original filter taps correspond to the HEVC legacy implementation [KAA+13] and the modified filter taps correspond to the proposed method to reduce the complexity.

| | $\alpha = 1/4$ | $\alpha = 1/2$ |
|---|---|---|
| Original | -1, 4, -10, 58, 17, -5, 1 | -1, 4, -11, 40, 40, -11, 4, -1 |
| $N_h = 7$ | -1, 4, -10, 58, 17, -5, 1 | -1,4, -11, 40, 40, -11, 3 |
| $N_h = 5$ | 1, -9, 58, 17, -3 | 2, -10, 41, 37, -6 |
| $N_h = 3$ | -6, 58, 12 | -7, 42, 29 |
| $N_h = 1$ | 64 | 64 |

**Table 6.3:** *Luminance interpolation filters : original and modified with varying size*

| $\alpha$ | Legacy | $N_h = 1$ | $N_h = 2$ | $N_h = 3$ |
|---|---|---|---|---|
| 1/8 | -2, 58, 10, -2 | 64 | 57, 7 | -3, 61, 6 |
| 1/4 | -4, 54, 16, -2 | 64 | 49, 15 | -5, 56, 13 |
| 3/8 | -6, 46, 28, -4 | 64 | 41, 23 | -7, 50, 21 |
| 1/2 | -4, 36, 36, -4 | 64 | 32, 32 | -7, 41, 30 |

**Table 6.4:** *Chrominance interpolation filters : original and modified with varying size*

The overall approximation process is controlled by a parameter called *Approximation level control* as described in Figure 6.9. It sets the number of taps of the MC interpolation filters. The different filter categories: low, middle, intermediate and high and summarized in Table 6.5.

| Configuration | Chrominance filter size | Luminance filter size |
|---|---|---|
| low | 1 | 1 |
| middle | 2 | 3 |
| intermediate | 3 | 5 |
| high | 4 | 7 |

**Table 6.5:** *Filter size per configuration*

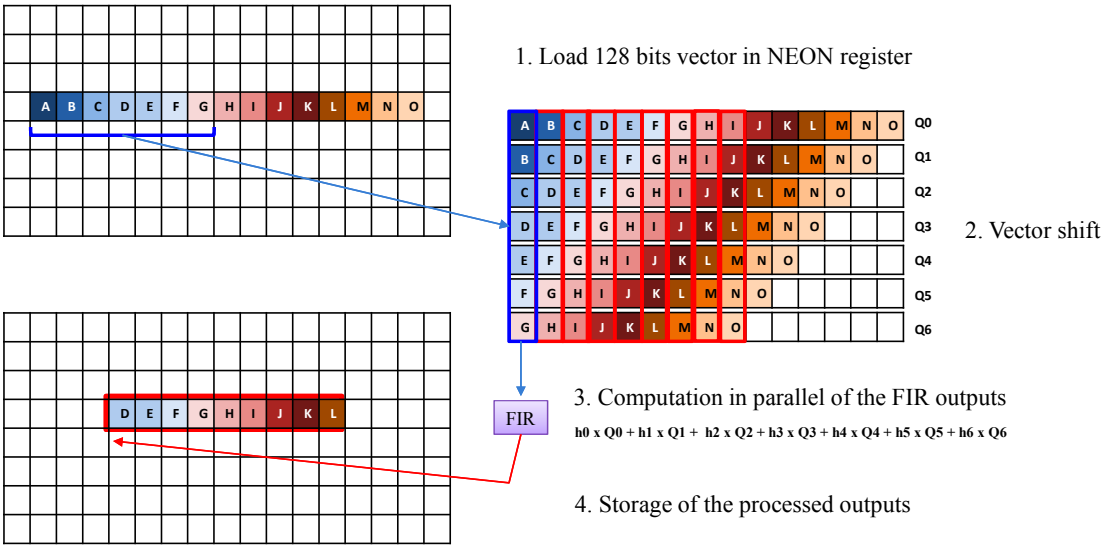


**Figure 6.10:** *Optimization using NEON methodology. 1. Load the NEON vector - 2. Shift the vector by step of one sample - 3. Compute the FIR output - 4. Store the processed outputs*

### SIMD Optimization of the Filters

One can argue that the filter size reduction brings speed-up for sequential filter implementations but may not be obvious with optimized ones. For systems relying on C-code only, the efficient implementation can use macro as shown in the example 6.1. To filter images, the height and width are processed successively.

```
#define LEGACY_FILTER(src, stride)                    \
    (filter[0] * src[x - 3 * stride] +  \
     filter[1] * src[x - 2 * stride] +  \
     filter[2] * src[x -     stride] +  \
     filter[3] * src[x               ] +  \
     filter[4] * src[x +     stride] +  \
     filter[5] * src[x + 2 * stride] +  \
     filter[6] * src[x + 3 * stride] +  \
     filter[7] * src[x + 4 * stride])

#define TAP3_FILTER(src, stride)                      \
    (filter[0] * src[x -     stride] +  \
     filter[1] * src[x               ] +  \
     filter[2] * src[x +     stride])
```

**Listing 6.1:** *Filter implementation of the legacy filter and 3-tap version*

Our proposal is based on *OpenHEVC* [Ope] where SIMD optimization is used in the filtering part. Data parallelism brings here execution speed-up. Figure 6.10 depicts how the data parallelism is used in the implementation. The methodology used is done in four steps. First, the 8-bit pixels are loaded into the available NEON vectors. Then the vector is shifted to represent the delay by one sample to represent the different time instances. The FIR outputs are processed in parallel where the FIR taps are loaded into the available registers.

The associated assembly code is given in the listing 6.2 and shows how the different vectors are filled. Vectors d16 to d22 load the rows offset by one sample $d_{16+i} = d_{16} >> i$.

```
//          a     b    c     d     e     f    g  reserved
// input d16   d17 d18 d19   d20   d21 d22   d23
.macro qpel_filter_1 out=q7
        vmov.u8   d24, #58
        vmov.u8   d25, #10
        vshll.u8  q13, d20, #4      // 16*e
        vshll.u8  q14, d21, #2      // 4*f
        vmull.u8   \out, d19, d24   // 58*d
        vaddw.u8  q13, q13, d20     // 17*e
        vmull.u8  q15, d18, d25     // 10*c
        vaddw.u8  q14, q14, d21     // 5*f
        vsubl.u8  q12, d22, d16     // g - a
        vadd.u16   \out, q13        // 58d + 17e
        vshll.u8  q13, d17, #2      // 4*b
        vadd.u16  q15, q14          // 10*c + 5*f
        vadd.s16  q13, q12          // - a + 4*b + g
        vsub.s16   \out, q15        // -10*c + 58*d + 17*e -5*f
        vadd.s16   \out, q13        // -a + 4*b -10*c + 58*d + 17*e -5*
            f
.endm
```

**Listing 6.2:** *Assembly version of the interpolation filter (legacy 7-tap implementation)*

The legacy filter kernel listed as example took 15 clock cycles, the new one with 3-taps takes 7 clock cycles. Consequently, with this new filter, the execution time is reduced and the decoding will need less energy thanks to DVFS and DPM techniques.

```
//          a     b     c
// input   d16   d17   d18
.macro qpel_filter3_1 out=q7
        vmov.u8   d24, #58
        vmov.u8   d25, #14
        vshll.u8  q13, d16, #3     // 8*a
        vmull.u8  \out, d17, d24   // 58*b
        vmull.u8  q15, d18, d25    // 14*c
        vadd.u16  \out, q15        // 58*b + 14*c
        vsub.s16  \out, q13        // 58*b + 14*c - 8*a
.endm
```

**Listing 6.3:** *Assembly version of the interpolation filter (3-tap modified implementation)*

Finally it shall be noted that an adequate optimized implementation can be found whatever the type of initial parallelism that is used. It confirms that execution speed-up can be achieved by reducing the number of processed samples even on data-paralleled implementations.

### 6.5.3 Computation Skipping of the Motion Compensation Filters and In-loop Filters

Another alternative for signal-oriented blocks with domain conservation is to skip processing as depicted in Figure 6.4. Because the error may be high, the block can be either totally skipped or skipped only periodically. A balance must be struck between QoS and computational complexity. A parameter called *skip control* is proposed that dynamically activates the processing block. All processing blocks that are classified as *signal-oriented* blocks with domain conservation can be modified with this approximated computing ap-

proach, namely the in-loop filters and the MC interpolation filters. Figure 6.11 shows an example of such implementation of the in-loop filters.

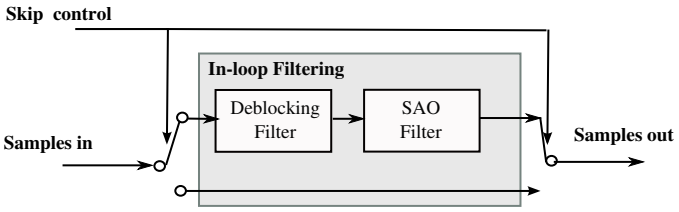


**Figure 6.11:** *In-loop filter skip*

The *skip control* parameter provides tuning of the video distortion at the decoder side. A decision is taken at the frame level to activate or not the filters. In an approximate computing system, this tuning capability is used to stay in the acceptable area of the application QoS for a given use case. This acceptable area is user-defined. It is represented by the grey area in Figure 6.1. The frequency of block skipping is set as a percentage of frames where filters are skipped. It leads to a fine quantum of quality distortion. By setting the *skip control* parameter to 0%, the decoder is similar to the legacy HEVC. If the *skip control* is set to 100%, the in-loop filters and the MC interpolation filters of chrominance and luminance are skipped permanently.

Four *skipcontrol* configurations are selected and categorized as shown in Table 6.6. These configurations are used in the rest of the study.

| Configuration | Block skipping percentage |
|---|---|
| low | 89% |
| middle | 63% |
| intermediate | 25% |
| high | 8% |

**Table 6.6:** *Percentage of block skipping per configuration*

## 6.6 Experiments

With a slight quality distortion, energy reduction when using the presented filtering techniques is measured on modern multicore hardware. One mobile and one desktop platform are chosen in order to compare the feasibility of the filtering methods on a wide range of use-cases.

### 6.6.1 Experimental Set-up

As a starting point, *OpenHEVC* [Ope] as reference software [CPG+13] is used here and is modified with the functionality presented in the previous sections. While we acknowledge that more optimized versions of the decoder exist [BBSF12], our intention is to compare the legacy implementation to the modified decoder in terms of energy savings on general purpose platforms.

The energy is measured as the average power dissipation for a fixed time execution, and the energy reduction is therefore equal to the average reduction in average power dissipation.

### Exynos 5410

The power measurements are conducted on two different hardware platforms. Similarly the previous chapters, we use the octa-core Exynos 5410 SoC as this SoC is widely used in recent smart phones and tablets [exy13].

Using the same methodology as the previous chapters, the power dissipation of the ARM platform is measured by reading internal power registers in the SoC while running the application under evaluation.The registers contain values representing the dissipated power for the Cortex-A15, A7, the external memory and the GPU measured in Watts. All values can be read from user space when issuing calls to specific control registers which handle the measurement sensors.

Listing 6.4 outlines the pseudo code for the power measurements using a shell script on the ARM platform.

```
loop over parameters{
   start_power_reading()
   start_HEVC() <parameters> <video>
   stop_power_reading()
   store_reading()
}
```

**Listing 6.4:** *Pseudo code for power measurements on ARM platform*

The additional workload of executing the power measurement script is measured to less than 1% and was hence considered neglected in the experiments.

### Intel i7

Secondly a quad-core desktop CPU based on the Intel i7-3770 with a clock frequency range between 1.6 GHz and 3.4 GHz is analyzed. This platform is widely used in normal consumer and server PCs. The hyperthreading and the Intel Turbo Boost were disabled for all experiments in order to increase the predictability and repeatability.

Since this platform does not include internal power registers, we added an external power measurement device connected to the ATX socket used as power supply of the CPU. The external device consists of a A/D converter connected to a Raspberry Pi device as illustrated in Figure 6.12. The INA226 A/D converter is capable of quickly reading both the current and the voltage of the input feed and perform the power computation. The converter then sends the computed power value on the i2c bus connected to an external device, here a Raspberry Pi, which stores all readings with a sample rate of 10ms.

Instead of executing power measurement scripts, the i7 sends a simple start signal over UDP to the Raspberry Pi before the benchmark starts and a stop signal after the benchmark is completed.

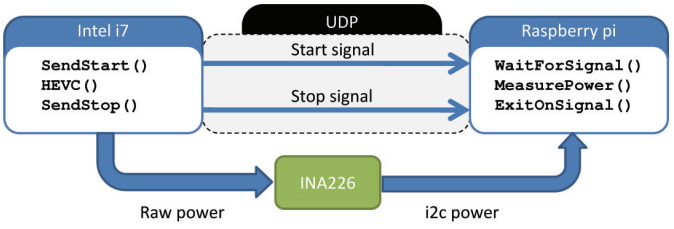


**Figure 6.12:** *External power measurement device connected to the Intel i7*

The ping latency of roughly 2ms was neglected since the execution time of benchmark is orders of magnitude larger.

### 6.6.2 Energy and Power measurements

A standard Ubuntu distribution for all experiments is used, and the *ondemand* [PS06] frequency governor is configured in all experiments as power management. The power measurements are obtained by running the HEVC decoder on four threads for a fixed number of frames and with various configurations and with various video types.

The parameters that are tested are: Input video sequence, Video QP, *ApproximationLevelControl* and *SkipControl*.

The same videos and configuration parameters were used on both platforms. The results are presented as averaged power in Watts. The efficiency of the proposed methods are reported as a saving relatively the legacy decoder as per Equation (6.8) in %.

$$saving = \left(1 - \frac{P_{proposed}}{P_{legacy}}\right).100 \tag{6.8}$$

with $P_{legacy}$ the average power consumption of the legacy decoder and $P_{proposed}$ the average power consumption of the modified decoder.

#### Exynos 5410

Table 6.7 and Table 6.8 present the results of the ARM platform. It can be noticed that the power consumption varies accordingly to the QP parameter and video content. The average power saving can be tuned according to the configuration parameter *ApproximationLevelControl* and *SkipControl* depending on the type of modified decoder.

#### Intel i7

In the same fashion as for ARM power results, Table 6.9 and Table 6.10 report the performance for the Intel platform. Similar characteristics can be observed: the proposed decoders can offer a fine grain tuning parameter to lower the power consumption of the video decoder.

In the next sections of the this chapter, an analysis of how this power reduction can be used is provided. Moreover, the behavior of the two proposed approximate computing techniques is detailed.

### 6.6.3 Energy Reduction and QoS Tradeoff exploration

The algorithmic-level approximate computing method proposes different approaches to reduce the energy consumption within a video system. The energy savings are obtained at the cost of a reduction of the decoding quality.

#### QoS Evaluation

##### *Objectives measurements*

The bitrate/distortion tradeoff is used as the system sizing variable. The QoS, expressed in PSNR or SSIM (see Section 6.4.5) can be degraded as long as it remains in the acceptable (grey) area of Figure 6.1. The PSNR tool is first used to quantify the relative performance of the proposed approximate computing performance.

| QP22 | Legacy | High | Intermediate | Middle | Low |
|---|---|---|---|---|---|
| *BQTerrace* | 6.725 | 6.085 | 5.058 | 4.534 | 4.069 |
| *Kimono* | 5.779 | 5.506 | 4.318 | 3.937 | 3.510 |
| *BasketBall* | 6.758 | 6.285 | 5.202 | 4.795 | 4.284 |
| *ParkScene* | 5.509 | 5.446 | 4.410 | 4.135 | 3.622 |
| QP27 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 5.487 | 4.817 | 3.962 | 3.468 | 3.090 |
| *Kimono* | 4.694 | 4.316 | 3.602 | 3.398 | 2.850 |
| *BasketBall* | 5.537 | 4.684 | 4.060 | 3.567 | 3.325 |
| *ParkScene* | 4.624 | 4.214 | 3.721 | 3.275 | 2.963 |
| QP32 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 4.082 | 3.947 | 3.292 | 2.947 | 2.500 |
| *Kimono* | 3.993 | 3.754 | 3.224 | 2.930 | 2.524 |
| *BasketBall* | 4.333 | 4.031 | 3.470 | 3.070 | 2.806 |
| *ParkScene* | 3.711 | 3.737 | 3.263 | 2.736 | 2.473 |
| QP37 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 3.790 | 3.644 | 3.060 | 2.665 | 2.299 |
| *Kimono* | 3.572 | 3.404 | 2.904 | 2.674 | 2.279 |
| *BasketBall* | 3.989 | 3.716 | 3.135 | 2.869 | 2.558 |
| *ParkScene* | 3.487 | 3.309 | 2.830 | 2.555 | 2.331 |
| **Average** | **4.704** | **4.400** | **3.776** | **3.441** | **3.068** |
| **Savings** | **-** | **6.5%** | **19.7%** | **26.9%** | **34.8%** |

**Table 6.7:** *Average power (in Watts) for measurements on ARM platform with different values of ApproximationLevelControl*

In Figure 6.13, an analysis of the performance of the *SkipControl* approach is displayed. Several percentages of *SkipControl* are compared. In Figure 6.14, the performance of the *ApproximationLevelControl* approach is analyzed for different filter sizes. In both cases, the performance trends are similar and both methods lead to a limited quality distortion compared to the legacy decoder. It also shows that these parameters can be used to finely tune the level of distortion.

For video applications, a loss of quality may be acceptable when the end-user evaluation is left to human perception. One can argue that the performance of the currently deployed standard, H.264, defines an acceptable zone. In Figure 6.13 and Figure 6.14, the H.264 performance are added for comparison. The approximated HEVC decoder is shown to remains better than the (previous standard) H.264 with all parameter values for bitrates lower than 2.5 MBytes. For higher bit-rates, the configurations can be tuned to perform better than H.264. For H.264/Advanced Video Coding, the JM reference software has been used [JMr]. Each test sequence is coded using twelve different bit rates. The quantization parameter $QP$ varies in the range of 20 to 42 using the method outlined in [OSS+12a].

### *Subjective measurements*

Although a subjective test campaign was not performed as in the state-of-art, test sequences were submitted to video experts through a MPEG workgroup paper [NRPM15b]. The

| QP22 | Legacy | High | Intermediate | Middle | Low |
|---|---|---|---|---|---|
| *BQTerrace* | 5.925 | 6.005 | 6.163 | 5.907 | 5.669 |
| *Kimono* | 5.779 | 5.520 | 4.706 | 3.739 | 3.510 |
| *BasketBall* | 6.758 | 6.501 | 6.144 | 4.929 | 4.284 |
| *ParkScene* | 5.509 | 5.778 | 4.835 | 3.966 | 3.622 |
| QP27 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 5.487 | 5.035 | 4.179 | 3.325 | 3.090 |
| *Kimono* | 4.694 | 4.430 | 3.646 | 3.157 | 2.850 |
| *BasketBall* | 5.537 | 5.157 | 4.427 | 3.542 | 3.325 |
| *ParkScene* | 4.624 | 4.242 | 4.050 | 3.122 | 2.965 |
| QP32 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 4.082 | 3.830 | 3.283 | 2.682 | 2.500 |
| *Kimono* | 3.993 | 3.702 | 3.149 | 2.685 | 2.524 |
| *BasketBall* | 4.333 | 4.153 | 3.566 | 2.915 | 2.806 |
| *ParkScene* | 3.711 | 3.856 | 2.930 | 2.762 | 2.473 |
| QP37 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 3.790 | 3.483 | 2.895 | 2.554 | 2.299 |
| *Kimono* | 3.572 | 3.355 | 2.877 | 2.511 | 2.279 |
| *BasketBall* | 3.989 | 3.698 | 3.134 | 2.652 | 2.558 |
| *ParkScene* | 3.487 | 3.220 | 2.713 | 2.592 | 2.331 |
| **Average** | **4.704** | **4.499** | **3.919** | **3.315** | **3.069** |
| **Savings** | **-** | **4.40%** | **16.7%** | **29.5%** | **34.8%** |

**Table 6.8:** *Average power (in Watts) for measurements on ARM platform with different values of SkipControl*

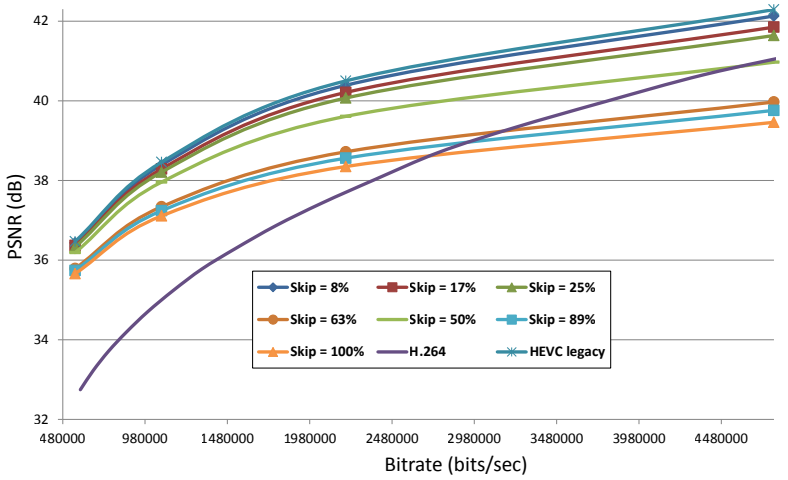


**Figure 6.13:** *PSNR degradation of approximate computing SkipControl HEVC decoder*

experts could check and evaluate subjectively the actual distortion induced by the proposed approaches.

| QP22 | Legacy | High | Intermediate | Middle | Low |
|---|---|---|---|---|---|
| *BQTerrace* | 18.215 | 16.088 | 15.061 | 14.432 | 13.954 |
| *Kimono* | 13.752 | 13.458 | 12.020 | 11.680 | 11.005 |
| *BasketBall* | 15.861 | 13.845 | 13.077 | 12.529 | 12.153 |
| *ParkScene* | 14.017 | 13.089 | 12.313 | 11.913 | 11.401 |
| QP27 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 12.788 | 11.963 | 11.121 | 10.555 | 10.150 |
| *Kimono* | 12.287 | 11.716 | 11.137 | 10.435 | 10.026 |
| *BasketBall* | 12.973 | 12.170 | 11.420 | 10.885 | 10.530 |
| *ParkScene* | 12.562 | 11.862 | 10.913 | 10.338 | 10.027 |
| QP32 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 11.386 | 11.070 | 10.296 | 9.753 | 9.323 |
| *Kimono* | 11.324 | 11.040 | 10.346 | 9.849 | 9.468 |
| *BasketBall* | 11.858 | 11.375 | 10.652 | 10.173 | 9.781 |
| *ParkScene* | 11.357 | 11.095 | 10.292 | 9.862 | 9.445 |
| QP37 | Legacy | High | Intermediate | Middle | Low |
| *BQTerrace* | 11.051 | 10.836 | 10.000 | 9.459 | 9.014 |
| *Kimono* | 10.929 | 10.753 | 9.910 | 9.473 | 9.141 |
| *BasketBall* | 11.263 | 10.912 | 10.207 | 9.757 | 9.365 |
| *ParkScene* | 10.702 | 10.833 | 9.884 | 9.404 | 9.032 |
| **Average** | **12.645** | **12.007** | **11.166** | **10.656** | **10.238** |
| **Savings** | **0%** | **5.1%** | **11.7%** | **15.7%** | **19.0%** |

**Table 6.9:** *Average power (in Watts) for measurements on Intel platform using ApproximationLevelControl decoder*

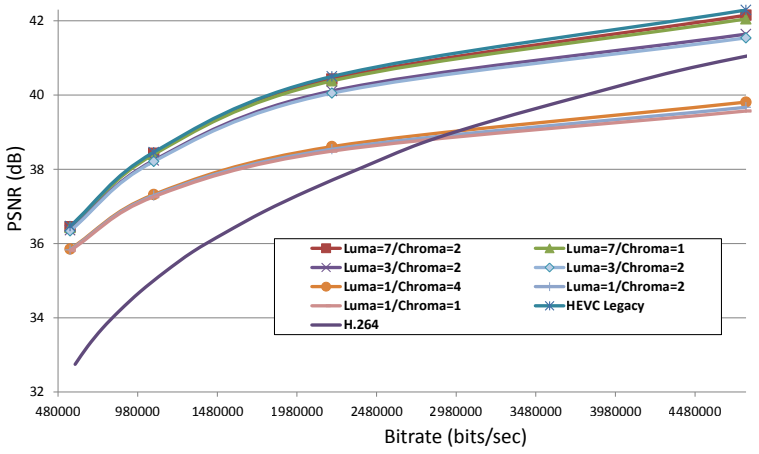


**Figure 6.14:** *PSNR degradation of approximate computing ApproximationLevelControl HEVC decoder. Luminance interpolation filter: nb of taps from 1 to 7 - Chrominance interpolation filter: nb of taps from 1 to 4*

The following is an excerpt of the working group meeting minutes from [FD15] (Section 2.2: Detailed Evaluation of Tunable Image Quality HEVC Decoding): "*The decoded bitstreams were viewed by the group and no artifacts were observed*".

| QP22 | Legacy | High | Intermediate | Middle | low |
|---|---|---|---|---|---|
| *BQTerrace* | 18.215 | 17.666 | 16.295 | 14.822 | 13.954 |
| *Kimono* | 13.752 | 13.411 | 12.582 | 11.414 | 11.005 |
| *BasketBall* | 15.861 | 15.231 | 14.008 | 12.763 | 12.153 |
| *ParkScene* | 14.017 | 13.578 | 12.908 | 11.661 | 11.401 |
| QP27 | Legacy | High | Intermediate | Middle | low |
| *BQTerrace* | 12.788 | 12.492 | 11.878 | 10.573 | 10.150 |
| *Kimono* | 12.287 | 11.680 | 11.371 | 10.420 | 10.026 |
| *BasketBall* | 12.973 | 12.658 | 12.106 | 10.859 | 10.530 |
| *ParkScene* | 12.562 | 12.006 | 11.697 | 10.354 | 10.027 |
| QP32 | Legacy | High | Intermediate | Middle | low |
| *BQTerrace* | 11.386 | 11.029 | 10.420 | 9.631 | 9.323 |
| *Kimono* | 11.324 | 11.016 | 10.500 | 9.838 | 9.468 |
| *BasketBall* | 11.858 | 11.563 | 10.877 | 10.116 | 9.781 |
| *ParkScene* | 11.357 | 11.276 | 10.498 | 9.777 | 9.445 |
| QP37 | Legacy | High | Intermediate | Middle | low |
| *BQTerrace* | 11.051 | 10.711 | 10.149 | 9.348 | 9.013 |
| *Kimono* | 10.929 | 10.609 | 9.952 | 9.376 | 9.141 |
| *BasketBall* | 11.263 | 10.965 | 10.344 | 9.663 | 9.365 |
| *ParkScene* | 10.702 | 10.526 | 9.839 | 9.330 | 9.032 |
| **Average** | **12.645** | **12.276** | **11.589** | **10.622** | **10.238** |
| **Savings** | **0%** | **2.9%** | **8.4%** | **16.0%** | **19.0%** |

**Table 6.10:** *Average power (in Watts) for measurements on Intel platform using SkipControl decoder*

Moreover it confirms that the PSNR metric is pessimist compared to the actual perception of the modified output. The main issue with PSNR is that a pixel shift can dramatically affect the performance whereas the quality can be equivalent. Because our proposed methods alter the motion restitution, the image can be shifted with no real impact on the quality. Therefore other metrics like SSIM reflect more accurately the actual human perception.

### Energy Reduction and Quality tradeoffs

The proposed decoders show good properties of energy consumption reduction. However, these energy savings introduce quality distortion. In Figure 6.16, the performance of the decoder with *computation approximation* of MC is depicted. The decoder is configured with four configurations as described in Table 6.5. It indicates that while the first 20% percent of saving can be easily be achieved, more savings come at the price of a high quality degradation. Figure 6.16 also shows that the performance knee arises for all configurations at approximately 25 %.

In Figure 6.17, the performance of the decoder with *computation skip* is depicted. Contrary to the decoder with *ApproximationLevelControl*, this proposal shows an interesting property of being of a close-to-linear form. As a consequence, the energy gains come at the same cost of quality regardless of the initial quality. This property can be used when


**Figure 6.15:** *Quality comparison between decodings of the same image using the modified decoder with Q22, the Legacy HEVC decoder QP22 and the Legacy decoder QP32.*

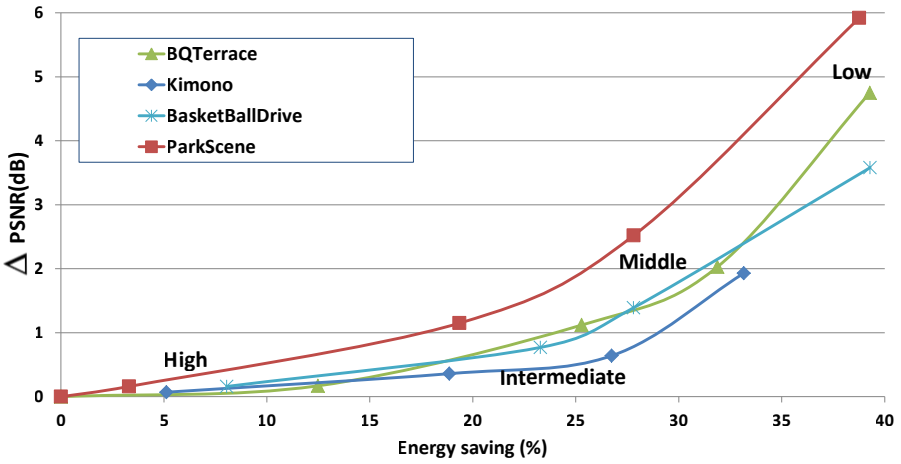


**Figure 6.16:** *PSNR degradation vs. energy saving of ApproximationLevelControl HEVC decoder*

the energy savings to obtain are high, the *ApproximationLevelControl* performing better than *ComputationSkip* for low energy savings. Additionally and to make the most of the two decoders, the two schemes could be merged to get a finer grain energy/quality tradeoff.

### Quality Scaling - an Alternative to QP Re-factoring

In modern systems where the video is displayed on portable devices, the challenge is to keep the device turned on as long as possible. The state-of-art mechanisms to reduce the decoding power consists in changing the QP of the incoming sequence. As a consequence, the quality of the decoded video is degraded.

Figure 6.18 and Figure 6.19 plot the power consumption trends as a function of the quality of the decoding. Here the SSIM indicator is analyzed as it fits better to the human perception. For a given quality, i.e. SSIM value, the purpose is to use the minimum power consumption. In other words, a Pareto curve of the system can be extracted.

In the same plot, the energy scalability of QP tuning is compared to the proposed *ComputationSkip* and *ApproximationLevelControl* methods. From a given QP, the re-

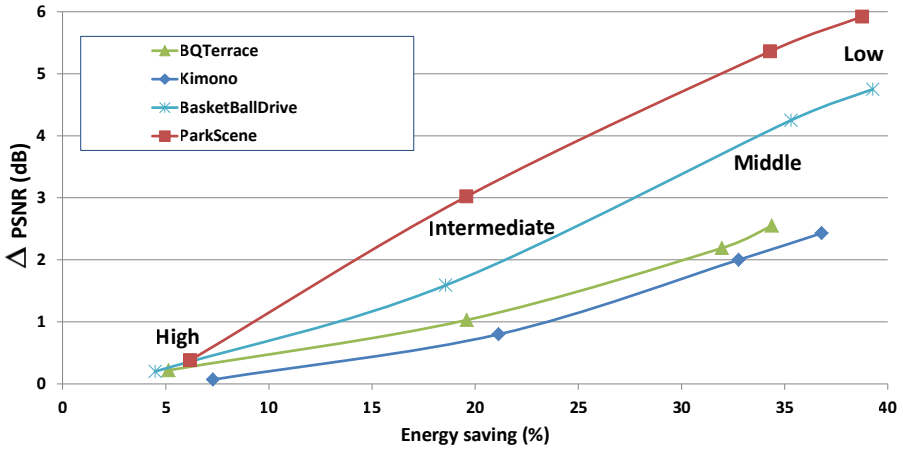


**Figure 6.17:** *PSNR degradation vs. energy saving of ComputationSkip decoder*

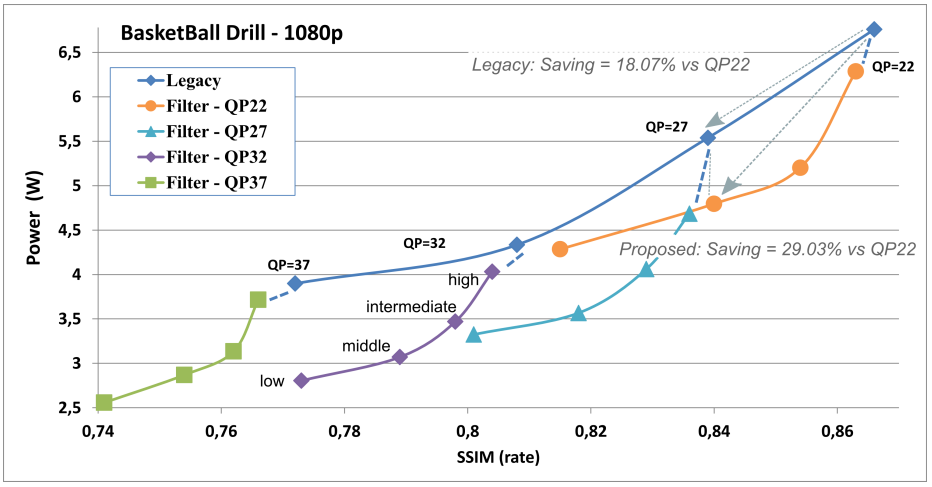


**Figure 6.18:** *Power consumption of ApproximationLevelControl decoder vs SSIM distortion*

sults show that increasing the QP improves the energy efficiency. The proposed decoders offer an alternative to reduce power consumption but providing a better tradeoff between energy efficiency and video quality.

In scenarios where the device is capable of choosing the QP like in DASH systems, the proposed methods can perform better results than changing QP at the source.

For example and assuming a reference video at $QP = 27$, changing the QP to $QP = 32$ can offer a reduction of 1 watt with a SSIM degraded from 0.84 to 0.8 (Figure 6.18). The decoder using *computation skip* can also achieve 1 watt of power reduction but with an SSIM reduced to only 0.82. With the decoder based on *computation approximation*, the SSIM can be only degraded to 0.83.

Besides, in broadcast scenarios where only one bitstream configuration is available, the proposed methods are the only possible ways that can be envisioned for saving energy on energy-limited devices. Each mobile device is then capable of adapting its own strategy without neither changing the input bitstream nor affecting the other devices. The maximum energy gains that are obtained on the test sequences is close to 40%.

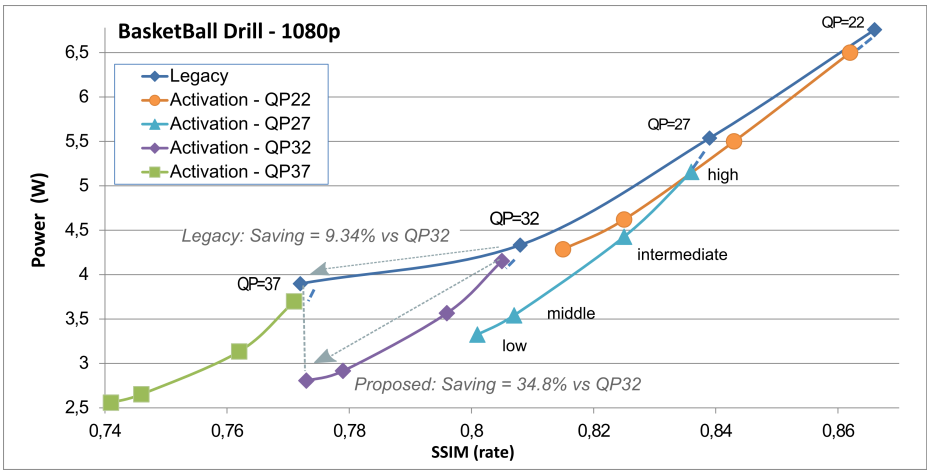


**Figure 6.19:** *Power consumption of ComputationSkip decoder vs SSIM distortion*

### Quality Drift

Video compression relies on the strong redundancy between frames, hence only motion are transmitted from a reference frame. As a consequence, one can argue that our proposal can induce a quality distortion that is not constant. A drift can happen due to error propagation between the reference frame and the predicted frames. Figure 2.4 recalls a frame dependency as shown in Section 2.2.2.

To get a complete picture of the impact on quality, a dedicated test sequence is generated to reflect different decoding scenarios. It concatenates the existing sequences *BasketBall Drive*, *Cactus* and *BQ Terrace* of the reference HEVC bitstreams [Bos12] as depicted in Figure 6.20.

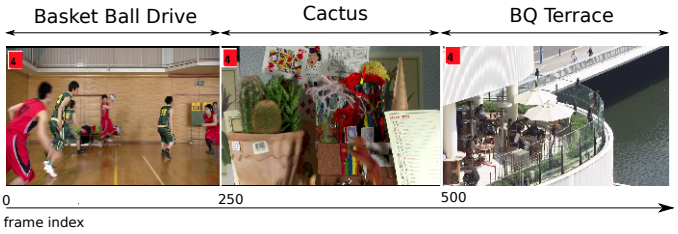


**Figure 6.20:** *Test sequence: concatenation of BasketBall Drive, Cactus and BQ terrace*

The chosen resolution is 1920x1080 and the bitrate of the sequence is set to 3.3 Mbits/s. The test bitstream is generated using the RA profile with a GOP of 25 frames.

Figure 6.21 depicts the performance of the configurations *Low*, *Middle*, *Intermediate* and *High*. For the sake of readability of the results, only the *BasketBall* sequence is presented of the *ComputationSkip* decoder. The same trend can be observed on all the sequences or with *ApproximationLevelControl* decoder.

As expected, the less complex the filters are, the more distortion there is in the decoded video. Besides, it shall be noted that the modified decoder introduces some performance degradation drift. However this drift is resynchronized at each decoded reference frame (e.g I-frame). The drift is due to the errors introduced by approximate filtering that are injected through inter-frame prediction in the depending images of the same GOP. Thanks to the resynchronization property, the PSNR cannot reach a dramatically low value over the sequence decoding. It shall be noted also that the PSNR drift is depending on the

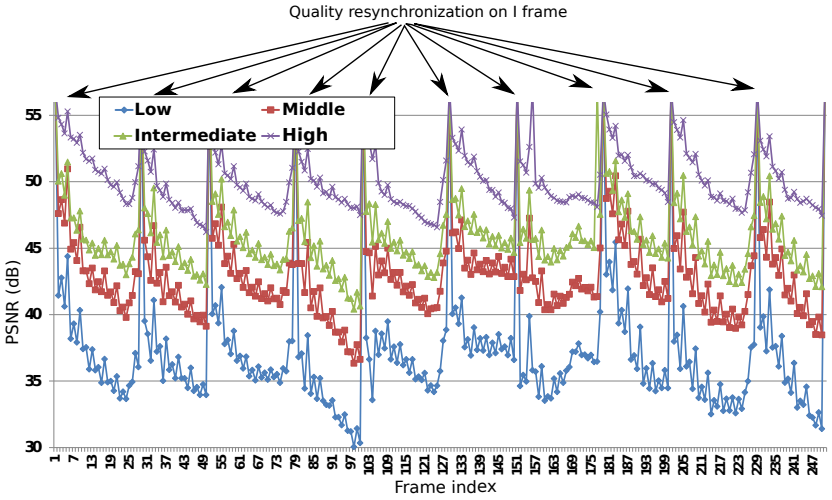


**Figure 6.21:** *PSNR per frame on the BasketBall sequence*

decoded frames. For example, between frames 80 to 100, the drift is high whereas it is barely null between frames 100 to 125. Therefore, an information of the expected drift could be appreciated to keep it under control or within the QoE requirements.

## 6.7 Proposal for Green Metadata Extension

In the previous experimental sections, we demonstrate that large power savings can be achieved on a SoC targeting handheld devices by using reduced filters, compared to a legacy HEVC decoder. However, the gains come at the cost of a quality distortion. This quality distortion depends on the scene content and can evolve over time.

The Green Metadata initiative within the MPEG standardization presented in Section 3.5 aims at offering means to the decoder to improve its energy efficiency by embedding metadata in the incoming bitstream. Because the quality distortion is depending on the scene content, a proposed metadata would consist in providing the potential PSNR distortion using modified HEVC filters.

In this case, the decoder could change dynamically the filters inside its decoding architecture in case energy needs to be saved and if the PSNR distortion is compatible with its QoS requirements. Figure 6.22 depicts an encoder architecture capable of generating metadata within the scope of the MPEG GreenMetadata initiative.

Compared to a legacy implementation, the encoder embeds the modified decoder with *approximate versions* of the filters. The reference output and the output of the modified decoder are sent to a Metadata generator. This block is in charge of computing the quality metric, e.g. PSNR, which is embedded into the GreenMetada container. Multiplexed to the legacy bitstream, a modified bitstream support is then sent to the transmitting system.

We proposed this new metadata to the GreenMetadata within several contributions [NRPM15b, RNPM15]. It is currently addressed by the group.

## 6.8 Conclusion

In this chapter, a design method that explores the tradeoffs between energy consumption and QoE in an application is proposed. It relies on the different algorithm-level techniques of approximate computing and the benefits are demoed for HEVC video decoders. For this

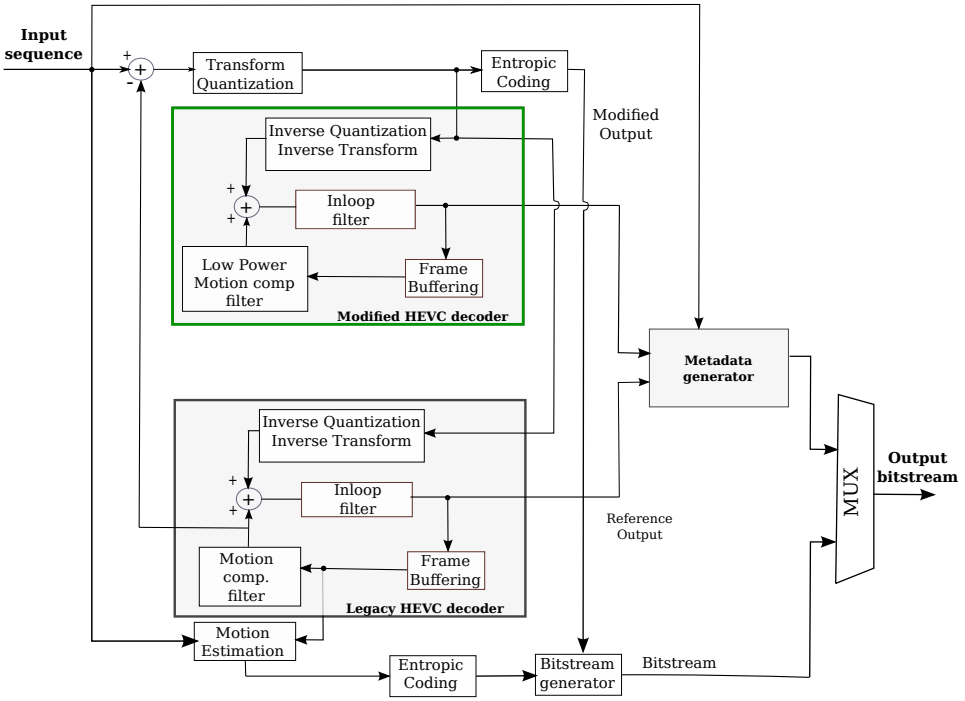


**Figure 6.22:** *Green Metadata proposal: Generation of a metadata to control quality and energy savings*

application, energy reductions of up to 40% are obtained with a moderate degradation of application quality. Moreover, parameters are obtained that can control the energy/distortion tradeoff at the decoder side. In terms of energy budgeting, algorithmic-level approximate computing widens the scope of application of standardized solutions. It can notably be used in conjunction with the other methods explored in the previous chapters.

However, the proposed schemes rely on a modification of the standard decoder. As of today, it is not allowed by standard bodies. For future versions of the standard, using approximate computing techniques as proposed in this chapter could be envisioned to design more energy efficient video decoders. This kind of study can input the standardization bodies for the "Future Video Coding" work item and proposes new profiles for post-HEVC.

# CHAPTER 7

# Conclusion

## 7.1 Summary

Over the last decade, electronic devices have gained support of ever more advanced features and their autonomy is under a great pressure. Among the new functionalities, the consumption of video-related contents has literally exploded together with the use of video sharing applications (e.g. Youtube, Netflix), social networks (e.g. Facebook, Instragram), IPTV on mobile and video-conferencing. Therefore, energy saving mechanisms applicable to video processing must be explored to support this increasing demand of battery supplied devices.

The contributions presented in this thesis address the challenge of developing signal processing applications, and more particularly cutting edge video decoders such as HEVC decoders, onto MPSoC based GPP. They show that the application characteristics (at different levels of knowledge) should be accounted for together with energy efficiency considerations. Significant gains can be obtained if low power techniques are used while knowing the applications needs. From a few tens of a Watt to a few Watts, the present average power of a single HEVC decoder on an MPSoC may be considered as negligible. However when summed up with the billions of devices that are bound to use video services in a near future, the impact of energy efficient design can be extensive at a global scale.

Table 7.1 summarizes the different studied items proposed in the thesis and classifies them with respect to the targeted low power techniques.

| | | DPM/DVFS joint optimisation | Approximate computing |
|---|---|---|---|
| Signal processing | Compile-time | 4.3 (monocore) / 4.4.2 (multicore) | 6.2, 6.3 |
| | Real-time | 4.2.3 | |
| HEVC decoding | Offline | 4.5.5 | 6.4, 6.5, 6.6 |
| | Real-time | 5.2.1, 5.3, 5.4, 5.5 | |

**Table 7.1:** *Contributions of the thesis per item*

In Chapter 4, a new energy model is introduced to take into account jointly the multicore capabilities of MPSoC and the low power hardware mechanisms such as DVFS and DPM. This model is inserted into a proposed rapid prototyping framework that gathers the MPSoC characteristics and the application performance to output an energy efficient scheduling under real-time constraints. The advantage of the proposed model is to formulate the energy efficiency problem into a single equation with DVFS and DPM parameters as variables. The model is also enhanced to support multicore architectures by integrating the parallelism level of application. Besides, the inherent convex property of the formulation is used to solve the optimization problem with either the *discipline convex programming* technique or the *geometric programming* technique. We show on various real-time scenarios that the proposed scheduling strategies identify the best operating point for each set-up. This best operating point always performs better than popular solutions such as *as-slow-as-possible* scheduling or *as-fast-as-possible* scheduling. The case study focuses on static signal processing applications, as stated in the scope of this chapter. Besides the proposed framework can be used for offline HEVC decoding. Experimental results are given on a embedded SoC and show energy gains of up to 87% over the state-of-the-art operating points.

In Chapter 5, the focus is moved to real-time HEVC decoding on MPSoC. Energy-saving mechanisms exploiting the DVFS of the underlying platform are proposed at two different layers. A particular attention is paid to the analysis of the benefits of exploiting application knowledge. The first proposal is a reactive scheme based on the actual real-time performance of the HEVC decoder. The performance information is reported to a DVFS control manager. This solution is not intrusive and only needs to know the real-time requirements at a system level. The second proposal is a proactive scheme based on the HEVC frame-type-based complexity predictor. This solution can be considered as intrusive. It needs to have a complete access of the application implementation. Both schemes present close-to-optimal energy performances with leading real-time performance. Low latency and low deadline-miss rates can be achieved.
In a nutshell, the main contribution of this chapter is the actual demo of the capability of achieving software based real-time decoding of state-of-the-art HEVC with low power considerations.

In Chapter 6, a modified HEVC decoder is proposed to address the ubiquitous video ecosystem. Indeed, video decoders can be embedded onto a wide range of devices with different decoding capabilities and QoS requirements. A framework based on approximate computing is applied at an algorithm level to provide a *modified* HEVC decoder. This modified decoder can improve the energy efficiency by up to 28% with a controlled quality distortion. We present in this chapter decoder modifications that can be done at different levels. Detailed implementations are also provided. They show that complexity reduction can be obtained either in C language or in assembly with NEON operators on ARM platforms. This latter implementation utilizes data parallelism for complexity reduction, improving then energy efficiency with a slight distortion. From a QoE point of view, the user experience of the proposed schemes is largely influenced by the screen size of the device. For small screens (e.g. mobiles devices), the proposal is barely noticeable. For large screens, the quality can be controlled through a metadata. Consequently, this proposal also addresses the adhoc working group at MPEG called GreenMetadata to assess further versions of the standard.

It must be noted that most of these contributions were either demoed or presented to the MPEG video community on top of the typical scientific contributions.

## 7.2 Future Work

The work presented in this thesis opens up opportunities for future research on energy efficiency or exploitation for the future of video codecs.

### 7.2.1 Energy modeling and Design Enhancement

#### 7.2.1.1 Enhanced Energy Modeling

The energy model introduced in Chapter 4 exhibits the potential gain brought by an efficient handling of the different power management techniques within MPSoC. However, in many cases, these techniques are only implemented partially on platforms. For example, deep sleep states can exist with different levels of sleep. In a future work, these different states could be taken into account to generate a finer grain representation and modeling. In this case, the challenge lies in keeping the convex property of the generated model. From a SoC design point of view, such model could be used to assist the design of ASIC sub-systems. Indeed the proposed framework targets signal processing applications. It defines the most appropriate parallelism level associated with a processing frequency. For specialized circuit design, a customized implementation could be done that targets energy efficiency on top of real-time requirements. Intensive filtering functions like video interpolation filters fall into this category of hardware Intellectual Property (IP).

#### 7.2.1.2 Support of Complex Applications

The model proposed in Chapter 4 assumes a streaming-like application. All the building functions have no built-in concurrency. This assumption comprises a wide range of applications especially in the signal processing area. However, to enlarge the scope of the target applications, future research can address the case of applications with competitive data paths. At first, this problem could be addressed with a partitioning methodology with time and data path. The objective would be to generate as many as needed sub-applications that would fulfill the assumptions formulated in Chapter 4. Therefore, each sub-application could be optimized from the energy-efficiency point of view.

### 7.2.2 Video Power Efficiency

#### 7.2.2.1 Video Encoding on Different Hardware Platforms

The offline decoding use case is assessed in Chapter 4. The proposed method shows how an energy efficient operating point can be found to process data when there is no real-time requirement. This operating point is not one of the obvious setting points. In future research, the same framework could be applied at the encoder side. Offline encoding is widely used to encode the incalculable number of video contents for recent VOD providers supported by streaming services. The power model of the hardware platforms that are used in data centers needs to be computed first. It can then input the proposed framework to generate the best energy efficient operating point and be compared to other strategies.

#### 7.2.2.2 Embedded DVFS Video Players

The video aware DVFS decoder presented in Chapter 5 shows that a pure software decoder can challenge low power real-time decoders. New standards such as HEVC could benefit from these implementations to take off. Indeed, with software-based implementation,

all the already-on-the-market devices can be upgraded to HEVC support with no hardware modification. This idea fits into the business model of Over-The-Top (OTT) service providers that leverage on streaming applications. However, using software-based solutions should be barely noticeable from the power supply point of view. For that purpose, video decoder shall embed advanced mechanisms as proposed in Chapter 5.

#### 7.2.2.3 New Interpolation Filters for Future Video Coding - An Alternative for Low Power Decoders

With the wide spread of video-capable devices, video decoders shall provide fine grain scalability of complexity. The proposed solution in Chapter 6 is to formulate less complex interpolation filters as they use a large share of the decoding complexity. This feature is not possible at the moment in HEVC. In a future mode, it could be possible for the terminal to adapt its filtering strategy to fit its decoding capabilities. Because handheld devices are energy-driven, low power modes in the decoder could be imagined using approximate computing techniques as proposed in Chapter 6.

#### 7.2.2.4 New Interpolation Filters combined with Approximate Computing Operators for ASICs

Chapter 6 emphasizes the great advantage of using algorithmic level approximate computing. Approximate computing also exists for circuits generating wrong results due to (controlled) incomplete computation. Several proposals exist in the literature for adders and multipliers that are the main operations supported in the interpolation filters. Because many hardware decoding solutions exist on the market based on ASIC technology, future research could analyze at first the impact of the proposed algorithmic level approximate computing for ASIC. Then, further research could combine low level approximate computing techniques with algorithmic level approximate computing techniques. This work could input the post-HEVC work item with MPEG.

# ANNEXE A

## French Summary

## A.1 Introduction

Aujourd'hui, les appareils électroniques offrent de plus en plus de fonctionnalités (vidéo, audio, GPS, internet) et des connectivités variées (multi-systèmes de radio avec WiFi, Bluetooth, UMTS, HSPA, LTE-advanced ...). La demande en puissance de ces appareils est donc grandissante pour la partie numérique et notamment le processeur. Pour répondre à ce besoin sans cesse croissant de nouvelles fonctionnalités et donc de puissance de calcul, les architectures des processeurs ont beaucoup évolué : processeurs multi-coeurs, GPU et autres accélérateurs matériels dédiés. Cependant, alors que de nouvelles architectures matérielles peinent à répondre aux exigences de performance, l'évolution de la technologie des batteries est quant à elle encore plus lente. Ainsi, l'évolution des batteries n'a pas encore permis une rupture technologique suffisante pour pallier la demande actuelle en énergie. En conséquence, l'autonomie des systèmes embarqués est aujourd'hui sous pression.

Parmi les nouveaux services supportés par les terminaux mobiles, la vidéo prend une place prépondérante. En effet, des analyses récentes de tendance montrent qu'elle représentera 70% du trafic internet mobile dès 2016. Accompagnant cette croissance, de nouvelles technologies émergent permettant de nouveaux services et applications. Parmi elles, HEVC permet de doubler la compression de données tout en garantissant une qualité subjective équivalente à son prédécesseur, la norme H.264. Cela permet d'élargir le spectre des applications possibles. De plus, le groupe de standardisation MPEG au sein de l'ISO a lancé des travaux visant à favoriser des implantations efficaces en consommation d'énergie de décodeurs vidéo [Gre13]. De ce cadre, le décodeur peut notamment adapter la qualité de la vidéo décodée en fonction de l'énergie nécessaire.

### A.1.1 Sources de consommation sur les systèmes embarqués

La consommation d'énergie dans les puces électroniques [Mud01] est souvent séparée par l'industrie CMOS en trois parties décrites dans la figure A.1.

Ils décrivent l'essentiel des caractéristiques pour les concepteurs de circuits logiques, les architectes et les constructeurs de systèmes. La consommation d'énergie est donnée par l'équation (A.1) [Mud01]. La première composante est la puissance dynamique $P_{Dyn}$

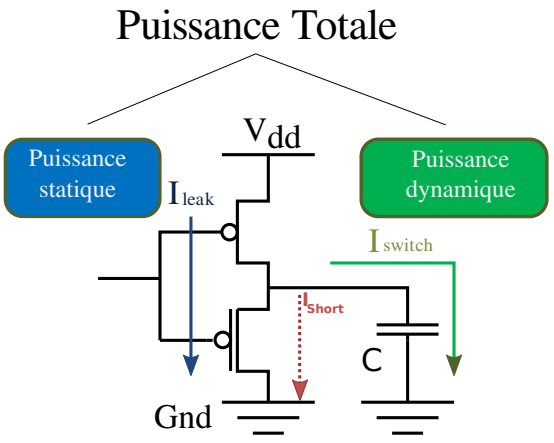


**FIGURE A.1** – *Consommation : trois composantes*

et résulte de la charge et la décharge capacitive sur la sortie de chaque porte logique. La deuxième composante $P_{short}$ représente la puissance dépensée en raison du courant de court-circuit, $I_{short}$. Le troisième élément mesure la puissance perdue par le courant de fuite quel que soit l'état de la porte.

$$P_{tot} = P_{Dyn} + P_{short} + P_{static} = \alpha.C.F_{clk}.V_{DD}^2 + \tau.\alpha.V_{DD}.I_{short}.F_{clk} + V_{DD}.I_{leak} \quad \text{(A.1)}$$

avec :

- $F_{clk}$ la fréquence d'échantillonnage du système
- $V_{DD}$ la tension d'alimentation
- $\alpha$ l'activité du processeur
- $C$ la capacité à la sortie des portes logiques
- $I_{short}$ le courant de court-circuit

### A.1.2 Contexte de la vidéo en électronique embarquée

Récemment, un certain nombre d'études a été publié sur la réduction d'énergie liée au décodage vidéo que ce soit au niveau architectural de la définition des systèmes ou au niveau des applications elles-mêmes.

Tout d'abord, plusieurs méthodes [CCKL11, HRD14b, MKA+13] ont été proposées afin d'utiliser la capacité des plateformes matérielles sur lesquelles repose l'application décodage vidéo. Une première classe d'études cherche à utiliser la capacité des processeurs à pouvoir adapter leur fréquence de fonctionnement et leur tension d'alimentation aux besoins réels de l'application. Appelé "DVFS", ce mécanisme permet de réduire la puissance dynamique des circuits et se prête particulièrement bien au décodage vidéo. En effet, dans le cas de décodage vidéo en temps réel, la vitesse d'affichage est plus faible que la vitesse de décodage potentielle, générant un temps de repos entre le décodage et l'affichage. Ce temps de repos est utilisé pour faire des économies d'énergie. Conjointement à ces techniques, la parallélisation des traitements de décodage permet de dégager encore plus de temps de repos.

Sous un autre angle d'optimisation, d'autres méthodes consistent à adapter le couple consommation/distorsion en fonction du niveau d'exigence de l'utilisateur et aux ressources disponibles [HKG+13, NHP+14]. Par exemple, il est proposé soit d'utiliser une résolution plus faible permettant ainsi de faire des gains en consommation sur le décodage, soit de désactiver des parties de décodeur temporairement.

### A.1.3 Problématique de la thèse

Grâce aux avancées technologiques offertes par l'industrie des semiconducteurs, on s'aperçoit que des traitements autrefois réservés à des implémentations matérielles peuvent se réaliser entièrement en logiciel. Ils peuvent même viser des composants embarqués. Parmi ces traitements, la vidéo est probablement l'application qui croît le plus rapidement depuis une décennie. Le curseur s'est déplacé vers l'autonomie de ces composants afin de garantir des temps d'activité les plus importants possible. Parallèlement à cela, les dernières applications vidéo comme HEVC doivent relever le challenge de l'implémentation pour attaquer rapidement le marché et prospérer dans le temps. Dans cette thèse, plusieurs solutions sont proposées afin de trouver une bonne adéquation entre les performances offertes par les plateformes embarquées et les caractéristiques de l'application considérée, ici le décodage de contenus HEVC.

### A.1.4 Plan

Ce résumé reprend l'organisation des chapitres du corps de la thèse. La partie A.2 propose d'intégrer dans un flot de prototypage rapide des considérations de basse consommation. Il est proposé d'intégrer les techniques d'ajustement de dynamique de la fréquence de traitement ainsi que les techniques d'endormissement dans la modélisation d'un problème d'ordonnancement de fonctions de traitement du signal. Le système est contraint par des exigences temps réel. Dans la partie A.3, une implémentation temps réel d'un décodeur HEVC est présentée. Celle-ci montre des performances de basse consommation proche de l'optimal tout en garantissant les exigences minimales du système. Enfin, la partie A.4 propose de modifier les traitements coûteux en énergie que sont les filtres d'interpolation vers des versions moins plus économes. Cette partie analyse les résultats en terme de consommation ainsi que le compromis en terme de distorsion de qualité. Enfin, la partie A.5 conclut ce résumé.

## A.2 Prototypage rapide sur systèmes multicoeurs

Le prototypage rapide est un outil qui permet d'automatiser le portage d'applications vers des plateformes. Des outils comme PREESM [PDH+14] modélisent l'application par des diagrammes de flux de données afin de les porter sur des plateformes hétérogènes multicoeurs. Cependant l'ordonnancement des fonctions élémentaires à exécuter se fait en cherchant à maximiser le débit de données produites en sortie du graphe d'exécution. Hors pour un système temps réel, l'exigence consiste à réaliser le traitement dans le temps imparti. Le temps de repos peut donc être utilisé pour économiser de l'énergie.

### A.2.1 Modélisation de l'énergie pour un processeur à fréquence variable

La consommation énergétique d'un composant électronique peut s'exprimer à travers la puissance utilisée. Cette consommation peut s'exprimer en fonction de la fréquence de

traitement comme définie dans l'équation (A.2). Elle désigne l'énergie consommée par cycle d'horloge.

$$E_{cycle}(f_{proc}) = \frac{1}{f_{proc}} . \int_0^T P_{tot}(t)\,\mathrm{d}t \tag{A.2}$$

avec $P_{tot}$ indiquant la puissance totale et $f_{proc}$ la fréquence de traitement.

D'un point de vue plus formel, la minimisation de l'énergie d'un système temps réel peut se formuler comme un problème d'optimisation. Il consiste à minimiser la consommation globale d'énergie tout en garantissant l'exécution de l'application dans le temps imparti, ici, avant que l'échéance temps réel n'arrive. L'équation (A.3) formule ce problème d'optimisation. Revenant à l'approche sur le prototypage rapide, $f_{proc}$ est la variable à ajuster en fonction de la charge de travail à réaliser.

$$\begin{aligned} \underset{f_{proc}}{\text{minimiser}} \quad & E_{tot}(f_{proc}) \\ \text{sachant que} \quad & \frac{w_{proc}}{f_{proc}} \leq D \\ & f_{proc} \geq f_{min} \\ & f_{proc} \leq f_{max} \end{aligned} \tag{A.3}$$

avec $f_{proc}$ la fréquence de traitement en Hz, $w_{proc}$ la charge de travail de l'application en cycles, $D$ le temps maximum alloué en secondes, $f_{min}$ la fréquence minimale de traitement en Hz proposée par la plateforme et $f_{max}$ la fréquence maximale de traitement en Hz proposée par la plateforme.

Pour illustrer les modèles proposés ci-dessus, la plateforme Exynos 5410 est analysée. Elle est composée de deux sous-parties ("cluster" en anglais). La première est orientée basse-consommation et est faite de Cortex A7 de chez ARM. L'autre est orientée puissance de calcul et est composée de Cortex A15. La fréquence de fonctionnement de l'ensemble peut être paramétrée. La fréquence minimale est 250 MHz et la fréquence maximale est de 1600 MHz. La figure A.2 montre la puissance totale du système en fonction de son mode de fonctionnement.

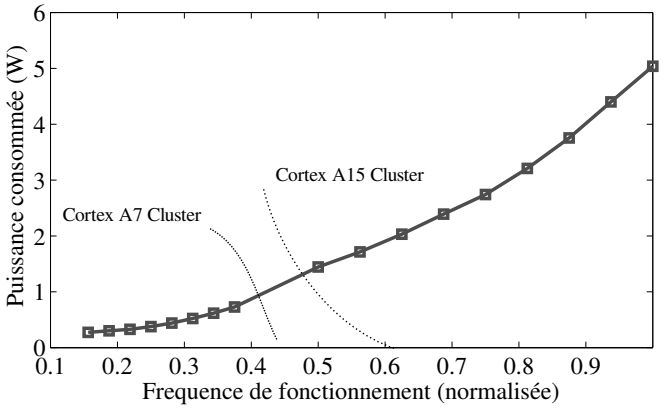


**Figure A.2** – *Puissance utilisée en fonction de la fréquence de traitement (normalisée)*

A partir de la caractéristique de puissance du système analysé (Figure A.2), l'équation (A.3) est utilisée pour définir le rendement énergétique. La figure A.3 montre les performances de rendement énergétique en fonction de la fréquence de fonctionnement. On peut noter en particulier l'apparition d'un minimum autour de 0.25. On peut aussi noter

que l'allure générale de l'énergie en fonction de la fréquence est convexe. Cette propriété peut être utilisée pour résoudre le problème de minimisation introduit par l'équation (A.3).

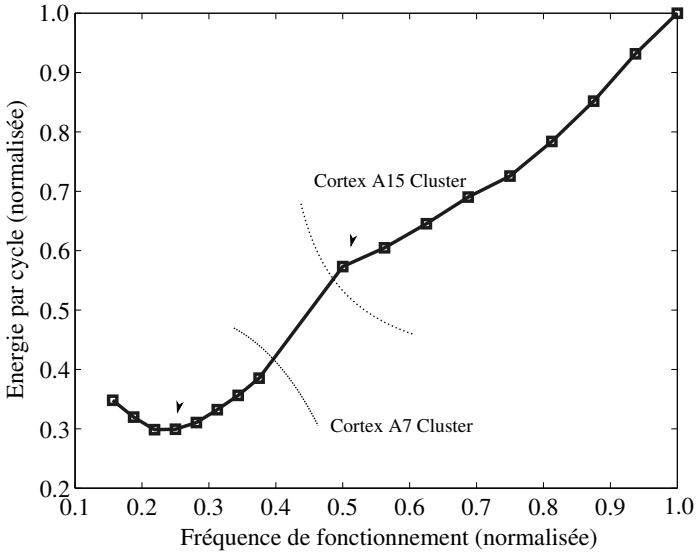


**Figure A.3** – *Rendement énergétique en fonction de la fréquence de traitement (normalisée)*

### A.2.2 Flux de travail

La figure A.3 a montré le caractère convexe de l'énergie. En utilisant un mode de résolution convexe dans l'équation (A.3), il est possible de trouver une solution au problème posé. Il existe plusieurs types d'outils de résolution. Ici, `CVX` [GBY08] est utilisé et intégré à un flot de développement visant à intégrer à la fois les caractéristiques de la plateforme (définition du rendement énergétique) et celles de l'application (contraintes temps réel). Afin d'obtenir une formulation mathématique des caractéristiques de la plateforme, la méthode des moindres carrés est utilisée pour identifier un modèle de la puissance consommée en fonction de la fréquence de traitement. L'équation (A.4) modélise la formulation

$$\begin{aligned} p_1 &= a_0 + a_1 f_{11} + a_2 f_{12} + ... + a_p f_{1q} + \varepsilon_1 \\ p_2 &= a_0 + a_2 f_{21} + a_2 f_{22} + ... + a_p f_{2q} + \varepsilon_2 \\ &.. \\ p_n &= a_0 + a_2 f_{n2} + a_2 f_{n2} + ... + a_p f_{nq} + \varepsilon_q \end{aligned} \tag{A.4}$$

Sous forme matricielle, l'équation (A.4) devient l'équation (A.5).

$$Y = Xa + \varepsilon \tag{A.5}$$

avec $P = \begin{pmatrix} p_1 \\ p_2 \\ .. \\ p_n \end{pmatrix}$, $X = \begin{pmatrix} 1 + .. + f_{1q} \\ 1 + .. + f_{2q} \\ .. \\ 1 + .. + f_{nq} \end{pmatrix}$, $a = \begin{pmatrix} a_1 \\ a_2 \\ .. \\ a_q \end{pmatrix}$, $\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ .. \\ \varepsilon_q \end{pmatrix}$

L'équation (A.5) peut être reformulée pour exprimer le vecteur de régression $\widehat{a}$ comme dans l'équation (A.6).

$$\widehat{a} = (X'X)^{-1}X'Y \tag{A.6}$$

Donc $\widehat{P} = f\widehat{a}$ devient l'estimation des mesures réelles de puissance $P$ à partir de différentes combinaisons de la fréquence de fonctionnement $F$.

Enfin la qualité de l'estimation est mesurée via le facteur de détermination $r^2$. Plus celui-ci est proche de 1, plus l'estimation est proche des mesures effectuées.

La figure A.4 représente le flux de travail qui est proposé. Le *développeur* injecte les mesures de puissance de la plateforme considérée ainsi qu'une liste de fonctions dites *convexes*. Le flux de travail va alors chercher une représentation satisfaisante des mesures, le tout sous la forme d'une équation convexe. Cette équation est alors rentrée dans le résolveur de problème convexe `CVX` qui se charge de trouver le point de fonctionnement optimal satisfaisant les contraintes de temps réel. L'étape de simulation permet de valider les résultats proposés par le résolveur `CVX`.

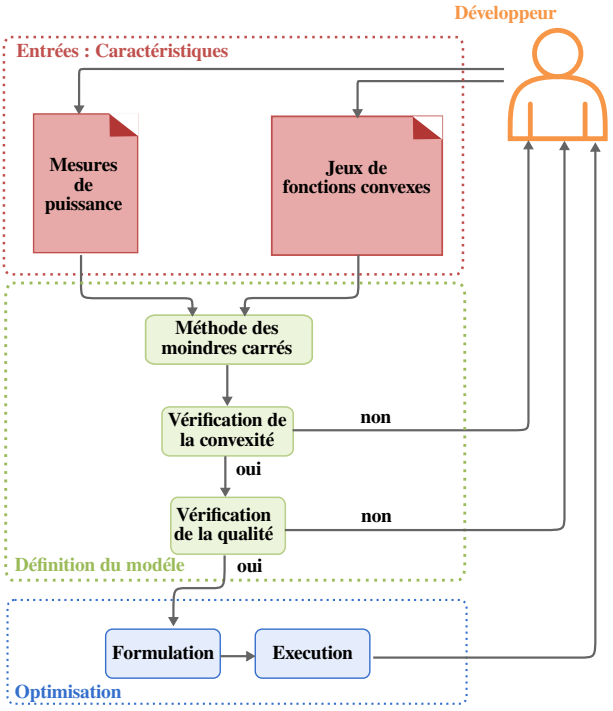


**Figure A.4** – *Flux de travail proposé au développeur*

### A.2.3 Évaluation du modèle

Dans cette section, le flux de travail est appliqué à une plateforme largement utilisée dans les systèmes embarqués. Il s'agit l'Exynos 5410 dont les caractéristiques sont rappelées dans la table A.1.

Pour illustrer le flux de travail présenté dans la figure A.4, le tableau A.2 donne la sortie des étapes de l'estimation au sens des moindres carrés pour quatre modèles.

Le tableau A.2 rapporte les mesures de qualité par modèle d'interpolation. Ces modèles sont des combinaisons de fonctions convexes. Dans la figure A.5, les quatre modèles sont comparés avec les mesures réelles. On peut noter que d'un point de vue qualitatif, le modèle 2 est le plus performant.

| | |
|---|---|
| SoC | Samsung Exynos5 Octa 5410<br>ARM Cortex-A15 Quad 1.6 GHz max.<br>ARM Cortex-A7 Quad 600 MHz max. |
| Memoire | 2GB LPDDR3 @ 800MHz |
| OS | Ubuntu 14.04<br>with GCC 4.8.2 |
| Capteurs d'énergie | INA 231<br>Fréquence de la mesure : 10 Hz |

**Table A.1** – *Configuration expérimentale*

| Modèle | Puissance($f_{norm}$) | $r^2$ |
|---|---|---|
| 1 | $p_0\sqrt{f_{norm}} + p_1 f_{norm}^2 + p_2 f_{norm}^4$ | 0.9984 |
| 2 | $p_0 f_{norm} + p_1 f_{norm}^2 + p_2 f_{norm}^3 + p_3 f_{norm}^4 + p_4 f_{norm}^5 + p_5 f_{norm}^6 + p_6 f_{norm}^7$ | 0.9999 |
| 3 | $p_0 f_{norm} log(f_{norm}) + p_1 f_{norm} + p_2 f_{norm}^3 + p_3 f_{norm}^5$ | 0.9916 |
| 4 | $p_0 f_{norm} + p_1 f_{norm}\sqrt{f_{norm}} + p_3 f_{norm} + p_3 f_{norm}^5$ | 0.9986 |

**Table A.2** – *Modèle d'interpolation de la puissance en fonction de la fréquence normalisée* $f_{norm}$

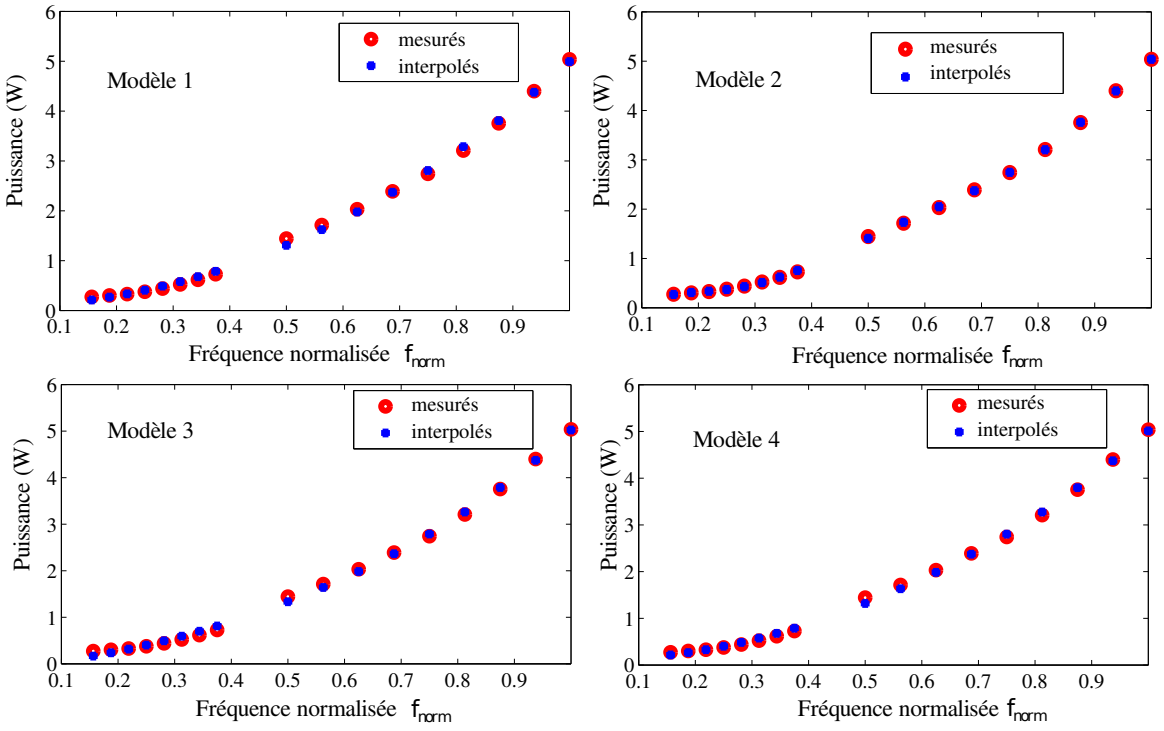


**Figure A.5** – *Comparaison de quatre modèles d'interpolation par rapport aux mesures de puissance,* $f_{norm}$ *étant la fréquence normalisée à 1600 MHz*

Cependant, l'analyse des coefficients dans la table A.3 montre que les modèles 2 et 4 ne remplissent pas exactement les critères élémentaires de convexité car ils sont négatifs pour certains d'entre eux. Donc seuls les modèles 1 et 3 peuvent être utilisés dans le flux de travail.

| Modèle | Valeurs des coefficient $p_i$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | p0 | p1 | p2 | p3 | p4 | p5 | p6 | p7 |
| 1 | 0.2620 | 4.4049 | 0.3278 | - | - | - | - | - |
| 2 | 44.3187 | -308.4337 | 1073 | -1958 | 1931 | -965 | 190 | -2.22 |
| 3 | 1.0197 | 2.9128 | 1.7585 | 0.3528 | - | - | - | - |
| 4 | 0.2029 | 6.7430 | -2.5781 | 0.6448 | - | - | - | - |

**Table A.3** – *Coefficients par modèle d'interpolation issu A.2*

### A.2.4 Discussions

Dans les sections précédentes, une méthode est détaillée permettant d'obtenir une modélisation fidèle de la consommation énergétique des composants électroniques. Grâce à ses propriétés de convexité, cette modélisation est intégrée dans un flux de développement. Les travaux présentés dans la thèse montrent également comment on peut généraliser la méthode proposée à des systèmes multicoeurs. Cette modélisation est alors utilisée pour prédire un point de fonctionnement optimal au sens de l'énergie pour les décodages hors-ligne de vidéo.

## A.3 Décodeur embarqué HEVC avec ajustement dynamique de la fréquence

Dans la partie précédente, le modèle présenté se concentre sur des applications de traitement de signal à temps d'exécution connu et fixe. Or le décodage en temps réel de vidéo ne respecte pas ces contraintes. Par construction, la complexité de décodage d'une trame est largement liée aux éléments qui la composent.

### A.3.1 Caractéristiques du décodeur video HEVC

A cause de l'utilisation d'algorithmes de traitement du signal sophistiqués comme le décodeur entropique ou la prédiction spatiale, les temps de traitement peuvent varier très largement d'une trame à l'autre. Comme le présente la figure A.6, les complexités de décodage peuvent être multipliées par un facteur 2 et plus. Adapter au plus juste la fréquence de traitement du processeur aux besoins en décodage de la séquence s'avère être un moyen très efficace de réduire la consommation énergétique du décodeur. Le défi est donc de pouvoir adapter cette fréquence sans enfreindre les contraintes de temps réel du décodeur. Les solutions de l'état de l'art se concentrent principalement dans l'utilisation d'un buffer de grande taille mais qui dégrade les performances globales du système [CAMJ14].

### A.3.2 Architecture de la proposition

Il est proposé alors de trouver la meilleure adéquation entre la fréquence de décodage et la fréquence d'affichage requise de la vidéo. Pour ce faire, la solution proposée cherche à

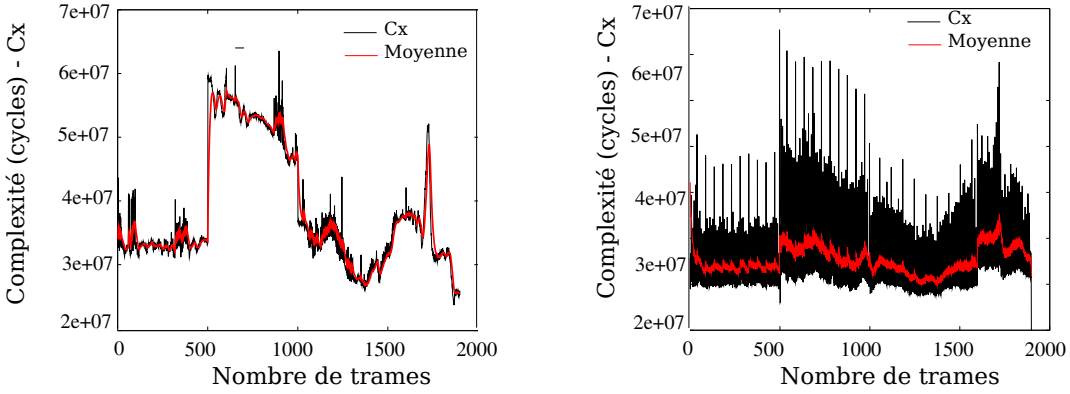


**Figure A.6** – *Exemples de complexité de décodage pour les profils avec prédiction spatiale (gauche) et avec prédiction temporelle (droite)*

adapter rétroactivement la fréquence en fonction de l'écart avec l'échéance temps réel de l'affichage. L'architecture haut-niveau est présentée dans la figure A.7.

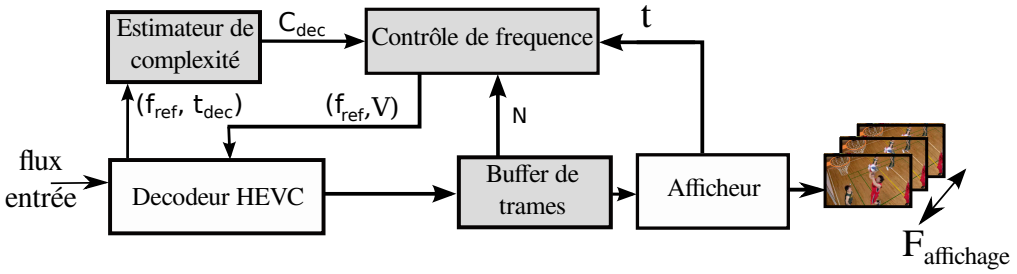


**Figure A.7** – *Architecture haut niveau de la solution proposée intégrant une gestion dynamique de la fréquence de fonctionnement*

Le décodeur vidéo utilisé est le décodeur *OpenHEVC* [Ope]. Il décode un flux d'entrée comprenant la vidéo compressée. En fonction du temps de décodage, le bloc *Estimateur de complexité* calcule le nombre de cycles nécessaires au décodage de la trame courante. Cette complexité est envoyée au bloc de *Contrôle de fréquence* qui adapte la fréquence d'affichage requise par la vidéo. Par ce biais, le but est de minimiser le temps de repos et donc la consommation de puissance dynamique. Le bloc *afficheur* est en charge de dépiler les trames décodeur à la fréquence requise d'affichage. S'il manque une trame, il le reporte au bloc de *Contrôle de fréquence* pour que la fréquence de fonctionnement soit accélérée. Le bloc *Buffer de trame* est inséré en mémoire tampon qui stocke les trames à afficher.

### A.3.3 Résultats

Les performances du système proposé sont évaluées suivant trois critères : la consommation énergétique, le taux d'échéance temps réel non-respecté et la taille du buffer tampon.

La même plateforme que dans la section précédente est utilisée. Ses caractéristiques sont résumées dans la table A.1. Pour avoir une vue complète des performances, un jeu de séquences de test référence [bbc] est utilisé. Leurs caractéristiques sont résumées dans le tableau A.4.

Tout d'abord, la table A.5 décrit les performances en terme de consommation de puissance. Elle compare notamment la solution proposée aux schémas classiquement utilisés [Bro13] et le schéma optimal. Ce schéma optimal consiste à décoder la vidéo à la fréquence

| Classe | Sequences | Résolution | Fréq. trame (Hz) |
|---|---|---|---|
| B | *Kimono (K)* | 1920x1080 | 24 |
| | *Cactus (C)* | | 50 |
| C | *BasketballDrill (BD)* | 832x480 | 50 |
| | *PartyScene (PS)* | | 50 |
| | *BQMall (BQM)* | | 60 |
| | *RaceHorses (RH)* | | 30 |
| | *Mixed (M)* | | 30 |
| E | *KirstenAndSara (K)* | 1280x720 | 60 |
| | *FourPeople (FP)* | | 60 |

**Table A.4** – *Séquences vidéo testées : paramètres et caractéristiques*

la plus faible possible qui respecte le temps total de la séquence. Ce schéma est optimal pour la consommation de puissance mais ne respecte pas le temps réel au niveau trame. On peut remarquer que la solution proposée est proche du schéma optimal et surclasse les méthodes classiques.

| | | 832x480 | | | | 1280x720 | 1920x1080 |
|---|---|---|---|---|---|---|---|
| Video | | M | RH | BD | BQM | K&S | K |
| Trames par sec. | | 30 | 30 | 50 | 60 | 60 | 24 |
| Performance | Puissance (W) | 1.19 | 1.41 | 1.65 | 1.88 | 2.76 | 3.51 |
| | Gain (%) | - | - | - | - | - | - |
| OnDemand | Puissance (W) | 0.49 | 0.66 | 0.89 | 1.18 | 1.73 | 3.15 |
| | Gain (%) | **58.7** | **53.6** | **46.2** | **37.5** | **37.3** | **10.3** |
| DVFS Optimal | Puissance (W) | 0.29 | 0.33 | 0.44 | 0.59 | 1.39 | 2.15 |
| | Gain (%) | **75.7** | **76.4** | **73** | **68.8** | **49.6** | **38.8** |
| DVFS Proposé | Puissance (W) | 0.31 | 0.37 | 0.54 | 0.81 | 1.39 | 2.2 |
| | Gain (%) | **74.3** | **74.1** | **67.6** | **57.1** | **49.6** | **37.4** |

**Table A.5** – *Comparaison de la puissance moyenne consommée en fonction du schéma associé au décodage*

Dans un second temps, les performances temps-réel sont analysées. Elles consistent à mesurer le taux d'échéances temps-réel râtées (*DMR*) en fonction de ta taille de la mémoire (ou *buffer* en anglais) de temporisation. Le tableau A.6 décrit les résultats par schéma utilisé. On peut noter que la méthode proposée propose des résultats tout à fait satisfaisants contrairement à la méthode optimale qui, même si elle est meilleure au sens de l'énergie, ne peut pas satisfaire les exigences temps-réel.

| | Buffer | 480p | | | | 720p | 1080p |
|---|---|---|---|---|---|---|---|
| Video | | M | RH | BD | BQM | K&S | K |
| Trames par sec. | | 30 | 30 | 50 | 60 | 60 | 24 |
| Performance | 8-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| OnDemand | 8-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4-slot | 0 | 0 | 0 | 0 | $< 0{,}01$ | 0,084 |
| DVFS proposé | 8-slot | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6-slot | $< 0{,}01$ | 0 | 0,04 | 0,05 | 0 | 0 |
| | 4-slot | 0,245 | 0,134 | 1,904 | 0,673 | 0,768 | 1,869 |

**Table A.6** – *Taux d'échéances râtées et taille du buffer tampon en fonction du schéma utilisé*

## A.4 Décodeur modifié HEVC

Dans cette partie, nous utilisons une structure standard du décodeur HEVC [SOHW12b]. Celui-ci se divise en plusieurs parties comme décrites dans la figure A.8. Dans un premier temps, le décodage entropique permet l'extraction des éléments de syntaxe de la séquence d'entrée. Ensuite, les données sont dé-quantifiées puis transformées par une Transformée en Cosinus Discrète (TCD) inverse. La prédiction est alors appliquée et peut être de deux types : soit *intra*, soit *inter* en fonction du type d'encodage de la séquence d'entrée. Dans le cas de la prédiction inter, elle est calculée en fonction des autres trames reçues et le décodage se fait grâce aux vecteurs de mouvement. Enfin, les filtres de boucle sont appliqués en fin de traitement pour réduire les éventuels artefacts et améliorer la qualité de l'image.

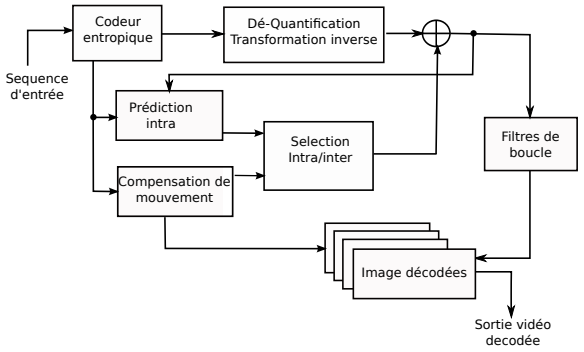


**Figure A.8** – *Version simplifiée du décodeur HEVC standard*

La réduction d'énergie peut se faire de plusieurs façons notamment si l'on autorise une légère distorsion de qualité. Lorsqu'on analyse les répartitions de charge du décodeur HEVC, on peut remarquer que les procédures de filtrage prennent une place prépondérante dans le total quelque soit le type de processeur utilisé [BBSF12, CPG+13]. Dans cette étude, nous proposons de modifier les caractéristiques des filtres utilisés afin de réduire la complexité de décodage, et donc, d'améliorer l'efficacité du décodage vidéo.

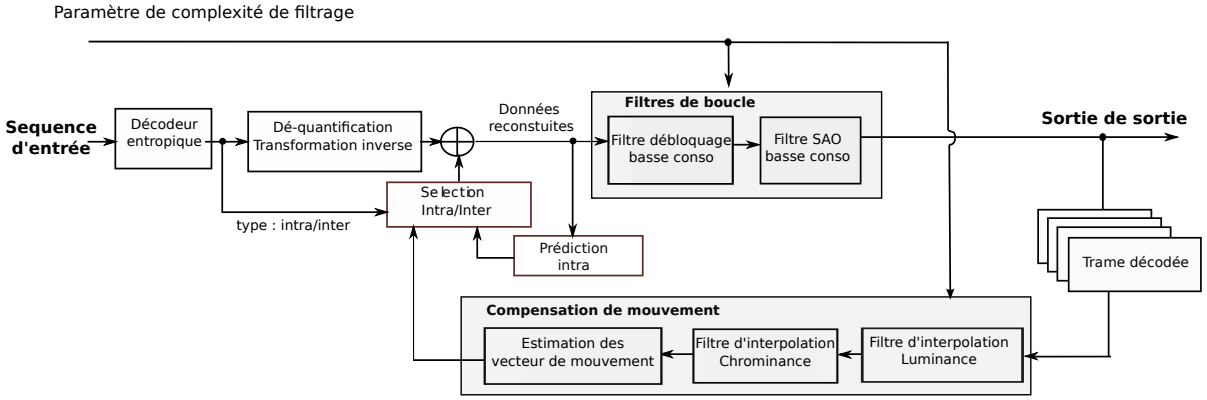


**Figure A.9** – *Décodeur HEVC modifié avec filtres basse consommation*

### A.4.1 Génération de filtres simplifiés

La proposition est donc réduire la complexité des filtres comme décrit dans la figure A.9. Cette réduction passe par la reformulation des coefficients de filtres. Dans la suite de l'étude, l'exemple des filtres d'interpolation est décrit.

Pour faire l'estimation de mouvements sur des fractions de pixel, des filtres d'interpolation 1-D sont utilisés dans HEVC. L'estimation de mouvement est utilisée par l'encodeur vidéo pour compresser la séquence à transmettre en exploitant la redondance temporelle entre les différentes trames de celle-ci. Un vecteur de mouvement est défini comme la position relative entre la trame à prédire et la trame de référence. Cependant, le mouvement des objets étant continu il ne peut correspondre exactement à des positions entières de pixel. C'est pourquoi en utilisant une estimation de mouvement fractionnaire (au sous pixel) l'estimation de vecteurs de mouvement est rendue plus précise et réduit l'erreur résiduelle. Kemal *et al.* analysent en détail la génération des filtres d'interpolation en fractions de pixel : soit $\{p_i\}(i = M_{min}, .., M_{max})$ of $Taille = M_{max} - M_{min} + 1$, la TCD permet d'obtenir le coefficient de Fourier $C_k$ (Eq. A.7). La paire transformation et transformation inverse peut être pré-calculée et fusionnée pour générer les positions fractionnaires [KAA+13].

$$C_k = \frac{2}{Taille} \cdot \sum_{l=M_{min}}^{M_{max}} p(l) cos(\frac{(2 \cdot l - 2 + Taille) \cdot k \cdot \pi}{2 \cdot Taille}) \tag{A.7}$$

Le standard HEVC utilise des filtres filtre à Réponse Impulsionnelle Finie (RIF) pour faire l'interpolation des pixels. Tseng *et al.* [TL08] proposent une méthode de synthèse de ces filtres à partir des coefficients de Fourier de l'équation (6.5). Ces filtres peuvent posséder un nombre pair ou impair de coefficients en fonction du type d'échantillons à interpoler : luminance ou chrominance. L'équation (A.8) donne l'exemple de génération de filtres pairs.

$$filtre_m(\alpha) = \frac{1}{M} cos(\pi \frac{m - \alpha}{N - 1}) \sum_{k=0}^{2M-1} (c_k^2 . \phi_k) \tag{A.8}$$

avec $\phi_k = cos(\frac{(2m-1+2M)\pi k}{4M}) cos(\frac{(2\alpha-1+2M)\pi k}{4M}))$.

Alors que le décodeur standardisé propose une configuration fixe pour chaque filtre, la solution proposée dans cette partie consiste à générer des filtres avec une taille variant de un coefficient jusqu'au nombre de coefficients du standard HEVC. Actuellement, les filtres

liés à la luminance utilisent $M_{min} = -3$, $M_{max}$ =4 et $Taille = 8$ alors que les filtres de la chrominance utilisent $Taille = 4$.
Dans cette étude, quatre classes sont définies relativement à leur complexité : faible, moyenne, intermédiaire et forte avec respectivement $Taille = 1, 3, 5, 8$ pour la luminance et avec $Taille = 1, 2, 3, 4$ pour la chrominance. Les tableaux A.7 et A.8 décrivent les filtres utilisés pour réduire la complexité.

| | $\alpha = 1/4$ | $\alpha = 1/2$ |
|---|---|---|
| Standard | -1, 4, -10, 58, 17, -5, 1 | -1, 4, -11, 40, 40, -11, 4, -1 |
| Taille=7 | -1, 4, -10, 58, 17, -5, 1 | -1,4, -11, 40, 40, -11, 3 |
| Taille=5 | 1, -6, 20, 54, -5 | 2, -9, 40, 40, -9 |
| Taille=3 | -4, 20, 48 | -9, 41, 32 |
| Taille=1 | 64 | 64 |

**Table A.7** – *Filtre d'interpolation pour la luminance : version standard et modifiée avec une taille variable*

| $\alpha$ | Standard | Taille=1 | Taille=2 | Taille=3 |
|---|---|---|---|---|
| 1/8 | -2, 58, 10, -2 | 64 | 58, 7 | -3, 62, 5 |
| 1/4 | -4, 54, 16, -2 | 64 | 50, 15 | -5, 58, 11 |
| 3/8 | -6, 46, 28, -4 | 64 | 41, 23 | -7, 51, 20 |
| 1/2 | -4, 36, 36, -4 | 64 | 32, 32 | -6, 42, 28 |

**Table A.8** – *Filtre d'interpolation pour la chrominance : version standard et modifiée avec une taille variable*

Afin de faciliter les comparaisons de performance, les différents filtres sont rassemblés en catégories et décrits dans le tableau A.9.

| Configuration | Taille des chroma | Taille des luma |
|---|---|---|
| faible | 1 | 1 |
| moyenne | 2 | 3 |
| intermédiaire | 3 | 5 |
| forte | 3 | 7 |

**Table A.9** – *Taille des filtres associée aux configurations*

### A.4.2 Résultats expérimentaux

### A.4.3 Banc de mesures

Les performances du système sont évaluées sur une plateforme Exynos 5410. Celle-ci est communément utilisée par les terminaux mobiles car elle est basée sur la technologie ARM big.LITTLE développée pour les applications mobiles basse consommation à base de processeurs généralistes. Cette plateforme adapte notamment sa fréquence de fonctionnement à la charge des processeurs réduisant ainsi l'énergie consommée. Les mesures de consommation sont réalisées par des capteurs placés sur le SoC qui permettent de collecter la puissance des processeurs et la mémoire à une fréquence de 10 Hz.

### A.4.3.1 Séquences de test

Pour obtenir une vue complète de la solution proposée, une séquence de test dédiée est proposée permettant d'étudier l'impact de différentes scènes sur le codage. Elle concatène les séquences standardisées pour HEVC : *BasketBall Drive*, *Cactus* et *BQ Terrace* comme décrit dans la figure A.10.

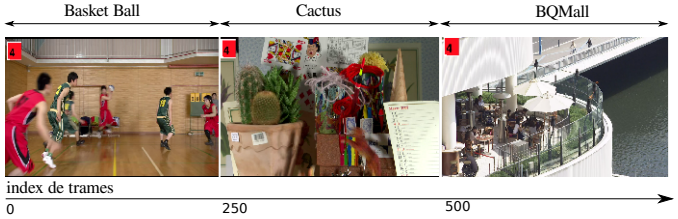


**Figure A.10** – *Séquence de test*

La résolution choisie est celle de la haute définition 1920x1080 avec un débit de 3.3 Mbits/s. L'image servant de référence à la prédiction est envoyée toutes les secondes à un rythme de 25 trames par seconde.

### A.4.3.2 Résultats

Dans cette section, la solution proposée est évaluée avec deux critères. Tout d'abord, les gains en énergie sont présentés. Puis, la distorsion par rapport à la solution standard est donnée.

La table A.10 résume l'énergie nécessaire pour décoder la séquence de test en fonction du mode de décodage. On peut alors mesurer les gains en énergie réalisés par la solution proposée par rapport à la solution standard en fonction du mode de filtrage.

| Configuration | Energie (J) | Gains (%) |
|---|---|---|
| Standard | 88.02 | - |
| basse | 63.02 | 28.37 |
| moyenne | 69.89 | 20.57 |
| intermédiaire | 72.37 | 17.79 |
| grande | 80.96 | 7.99 |

**Table A.10** – *Gains en énergie*

Les mesures montrent que des gains allant jusqu'à 28 % peuvent être atteints sur l'énergie nécessaire pour décoder la séquence. On remarque que les gains sont aussi directement liés à la complexité de filtrage choisie.

Dans un second temps, la qualité de décodage est estimée en calculant le PSNR trame par trame entre le l'image décodée par le schéma standard et la méthode proposée.

La figure A.11 trace le PSNR en fonction de l'index de trame décodée. On peut remarquer que la qualité est naturellement plus dégradée avec les filtres les moins complexes. Par ailleurs, on note aussi une dérive de la dégradation avec le temps et que cette dérive est stoppée à l'apparition d'une image de référence, toutes les 25 trames dans l'exemple. Cette caractéristique est intéressante pour la conception du système car elle permet au décodeur de maîtriser la dérive de qualité par ce mécanisme.

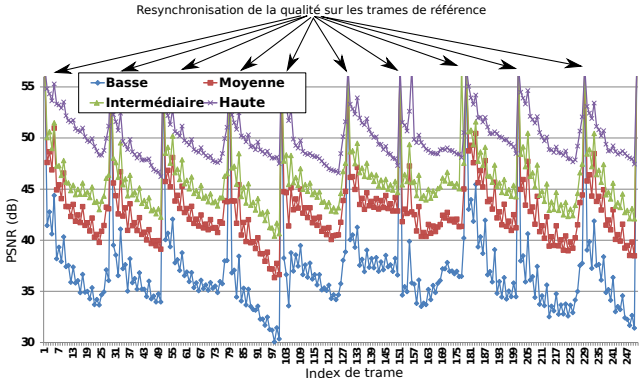


**Figure A.11** – *PSNR par trame sur la séquence Basket Ball drive*

## A.5 Conclusion

Les contributions dans cette thèse ont permis de proposer des solutions d'implémentation visant à chercher l'adéquation entre l'application vidéo et la plateforme. Ces contributions se placent aux différents niveaux d'abstraction de la conception d'un système comme il est présenté dans la figure A.12. Au final, ces travaux ont aussi permis de montrer qu'il est possible de développer des solutions compétitives de décodage video comme HEVC entièrement logicielles tout en étant basse consommation.

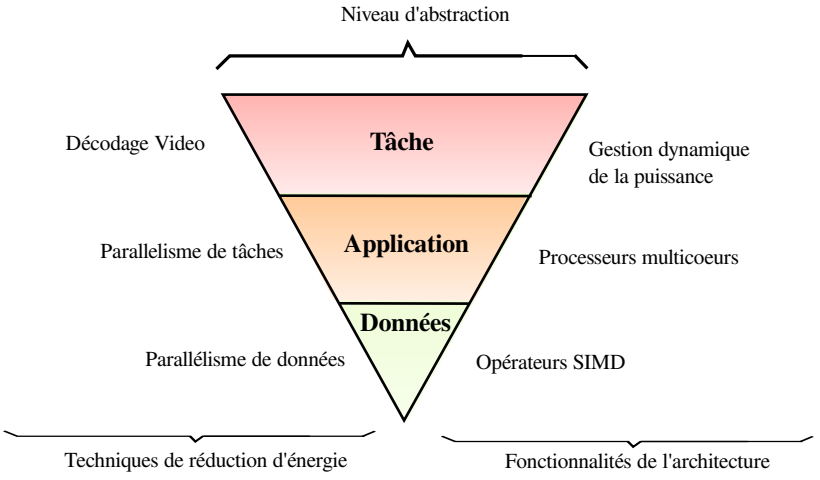


**Figure A.12** – *Optimisation énergétique en fonction du niveau d'abstraction*

### A.5.0.3 Perspectives

Le travail réalisé durant cette thèse a permis d'identifier un certain nombre d'opportunités pour de futurs travaux de recherche. Un premier axe serait de rendre possible l'identification du point optimal de fonctionnement pour des applications dynamiques. En effet, la méthode proposée est basée sur une connaissance de la charge de l'application. Par ailleurs, il serait aussi intéressant de combiner les méthodes proposées pour le décodage temps réel HEVC avec l'estimation en ligne des paramètres de consommation. Enfin, de futurs travaux de recherche pourront porter sur l'impact de la température sur la consommation de la plateforme.

# List of Figures

# List of Tables

# Acronyms

**AAC** Advanced Audio Coding. 118

**AAM** Algorithm-Architecture Matching. 51, 56

**ACPI** Advanced Configuration and Power Interface. 30

**ADVFS** Adaptive DVFS. 105, 107, 110, 112

**AI** All Intra. 83, 86, 123, 124

**ASIC** Application Specific Integrated Circuit. 37, 38, 147, 148

**AVX** Advanced Vector Extensions. 24

**BDVFS** Boosted DVFS. 101, 102, 105, 110, 112

**CABAC** Context-adaptive binary arithmetic coding. 16–18

**CAVLC** Context-adaptive Huffman variable-length coding. 17

**CCTV** Closed-circuit television. 12

**CISC** Complexed Instruction Set Computer. 24

**COFDM** Coded Orthogonal Frequency Division Multiplex. 13

**CTU** Coding Tree Unit. 25

**DASH** Dynamic Adaptive Streaming over HTTP. 6, 42, 140

**DCT** Discrete Cosine Transform. 16, 36, 123, 128

**DCTIF** DCT-based interpolation filter. 128

**DF** Deblocking filter. 123

**DMR** Dead Line Miss Ratio. 32, 111

**DPB** Decoded Pictures Buffer. 26

**DPM** Dynamic Power Management. 5, 27, 28, 38, 48, 49, 52, 55, 57, 61, 90, 92, 117, 120, 146

**DSIS** Double Stimulus Impairment Scale. 127

**DSP** Digital Signal Processor. 38, 124

**DTTV** Digital terrestrial TV. 12, 13

**DUT** Device-Under-Test. 82, 83, 85, 110

**DVFS** Dynamic Voltage Frequency Scaling. 5, 7, 27, 28, 30, 36, 38, 41, 48, 49, 52, 55–58, 61, 62, 69, 90–94, 100, 101, 110–112, 115, 117, 120, 146, 147, 150

**FHD** Full High Definition. 98

**FIR** Finite Impulse Response. 119, 128

**fps** frames per second. 32, 81, 102

**FSM** Finite State Machine. 49

**GOP** Group of Pictures. 19, 25, 41, 142

**GP** Geometric Programming. 49, 72, 73, 77, 85

**GPP** General Purpose Processor. 7, 8, 91, 92, 96, 98, 124, 145

**HDR** High Dynamic Range. 17, 18, 43

**HE-AAC** High-Efficiency Advanced Audio Coding. 118

**HEVC** High Efficiency Video Coding. 5–8, 15, 17, 18, 24–26, 32, 35, 37, 38, 42–44, 49, 55, 69, 90–92, 94, 95, 97, 98, 100, 108, 112, 113, 115, 117, 123–129, 132, 137, 145–149, 151

**HVS** Human Visual System. 8, 14

**IDR** Instantaneous Decoding Refresh. 120

**IIR** Infinite Impulse Response. 119

**Imain** Imain. 82, 95, 96

**IP** Intellectual Property. 147

**MA** Moving Average. 97

**MARS** Multivariate Adaptive Regression Splines. 62

**MC** Motion Compensation. 124, 128, 129, 132, 139

**MCSE** Methodology for the Conception of System of Electronics components. 47

**MMSE** Minimum Mean Square Error. 63, 64, 66, 72, 90

**MoC** Model of Computation. 48–50, 91

**MPEG** Motion Picture Expert Group. 5, 6, 8, 16, 115, 117, 118, 125, 128, 139, 142, 146

**MPSoC** Multiprocessor SoC. 6–8, 25, 26, 39, 47–49, 51, 52, 55, 56, 70, 72, 75, 76, 86, 145–147

**ODVFS** Optimal DVFS. 94, 100, 101, 107, 108, 110, 111

**OTT** Over-The-Top. 4, 147

**PAPR** Peak to Average Power Ratio. 13

**PSM** Power State Machine. 27, 28

**PSNR** Peak Signal To Noise Ratio. 126–128, 136, 139, 142, 143, 162

**PU** Prediction Unit. 25, 26

**QoE** Quality of Experience. 4, 7, 31, 42, 43, 146

**QoS** Quality of Service. 4, 7, 31, 33, 37, 47, 52, 115–122, 125–127, 132, 136, 146

**QP** Quantization Parameter. 36, 37, 95, 135, 140, 141

**RA** Random Access. 82, 84, 86, 95, 96, 123, 124

**RGB** Red Green Blue. 41

**RIF** filtre à Réponse Impulsionnelle Finie. 160

**SAO** sampling adaptive offset. 18, 123, 125

**SBR** Spectral Band Replication. 118

**SDF** Synchronous Data Flow. 49, 50

**SHVC** Scalable HEVC. 37

**SIMD** Single Instruction Multiple Data. 7, 24, 39, 92, 99, 122

**SoC** System-on-Chip. 6, 7, 20, 21, 29, 44, 94

**SRDF** Single-Rate Data Flow. 69

**SSE** Streaming SIMD Extensions. 24

**SSIM** Stuctured Similiraty. 126–128, 136, 139, 141

**SVC** Scalable Video Coding. 37

**TCD** Transformée en Cosinus Discrète. 159, 160

**VOD** Video On Demand. 32

**VQEG** Video Quality Experts Group. 127

**WPP** Wavefront Parallel Processing. 25

# Personal Publications

## Journals

[NMN+14] S. Holmbacka, E. Nogues, M. Pelcat, S. Lafond, J. Lilius and D. Menard, Energy-Awareness and Performance Management with Parallel Dataflow Applications In *Journal of Signal Processing Systems*, Springer, 2015.

[NMN+14] E. Nehmeh, D. Menard, E. Nogues, A. Banciu, T. Michel and R. Rocher, Fast Integer Word-length Optimization for Fixed-point Systems In *Journal of Signal Processing Systems*, Springer, 2014.

[RMM+15] E. Raffin, E. Nogues, W. Hamidouche, S. Tomperi, M. Pelcat and D. Menard, Low power HEVC software decoder for mobile devices In *Journal of Real-Time Image Processing*, Springer, 2015.

[NHR+16] E. Nogues, G. Herrou, L. Robin, M. Pelcat, D. Menard, E.Raffin and W. Hamidouche, Efficient DVFS for Low Power HEVC Software Decoder, **Submitted to the Journal of Real-Time Image Processing - Undergoing a Minor Revision**.

[NMP+16] E. Nogues,D. Menard and M. Pelcat, Algorithmic-level Approximate Computing Applied to Energy Efficient HEVC Decoding, **Submitted to the IEEE Transactions on Emerging Topics in Computing - Undergoing a Major Revision**.

## International Conferences

[DHP+14] E. Nogues, S. Holmbacka, M. Pelcat, D. Menard and J. Lilius, Power-aware hevc decoding with tunable image quality In *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS)*, 2014.

[NM+14] E. Nogues and D. Menard, Efficient fixed-point refinement of DSP dataflow systems, In *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS)*, 2014.

[HNP+14] S. Holmbacka, E. Nogues, M. Pelcat, S. Lafond and J. Lilius, Energy Efficiency and Performance Management of Parallel Dataflow Applications, In *Proceedings of the 2014 Conference on Design & Architectures for Signal & Image Processing*, 2014. **Best Paper Award**

[NRP+15] E. Nogues, E. Raffin, M. Pelcat and D. Menard, A modified HEVC decoder for low power decoding, In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, 2015.

[RHN+15] E. Raffin, W. Hamidouche, E. Nogues, M. Pelcat, D. Menard and S. Tomperi, Energy efficiency of a parallel HEVC software decoder for embedded devices, In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, 2015.

[NLP+15] E. Nogues, R. Lebidan and D. Pastor, Active Noise Control with digital PDM MEMS mics In *Proceedings of IEEE International Symposium on Consumer Electronics (ISCE)*, 2015.

[NBP+15] E. Nogues, R. Berrada, M. Pelcat, D. Menard and E. Raffin, A DVFS based HEVC decoder for energy-efficient software implementation on embedded processors, In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2015.

[NPM+16] E. Nogues, M. Pelcat, D. Menard and A. Mercat, Energy Efficient Scheduling of Real Time Signal Processing Applications Through Combined DVFS and DPM, In *Proceedings of 25st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2016. 7

[NPM+16] E. Nogues, M. Pelcat and D. Menard, Energy Optimization on MPSoC with Convex Framework, To presented at the *Design Automation Conference (DAC) - Work-In-Progress session*, 2016. 7

## Demo

[NLR+15] E. Nogues, M. Lacour, E. Raffin, M. Pelcat and D. Menard, Low Power Software HEVC Decoder Demo for Mobile Devices, In *USB Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2015. **Best Demo Award**

[RLN+15] E. Raffin, E. Nogues, M. Lacour,, M. Pelcat, D. Menard, K. Desnos and J. Nezan, Low Power Software HEVC Decoder Demo: A side-a-side Comparison, In *P Proceedings of the 2015 Conference on Design & Architectures for Signal & Image Processing*, 2015.

## Patents

[PPM+15] E. Nogues, M. Pelcat, D. Menard and E. Raffin, DECODEUR, PROCEDE ET SYSTEME DE DECODAGE DE FLUX MULTIMEDIA, PCT/EP2015/073964, October 2014.

[PPN+15] E. Nogues, M. Pelcat, D. Menard and E. Raffin, DECODEUR, PROCEDE ET SYSTEME DE DECODAGE DE FLUX MULTIMEDIA, France 1551085, February 2015.

## MPEG Technical work

[NRQ+15] E. Nogues, E. Raffin, M. Pelcat and D. Menard, HEVC Decoding with Tunable Image Quality - Subjective evaluation, Contribution to 111th MPEG meeting for GreenMetadata ad hoc group, 2015. 8

[NDR+14] E. Nogues, X. Ducloux E. Raffin, M. Pelcat and D. Menard, HEVC Decoding with Tunable Image Quality for Green Metadata applications, Contribution to 110th MPEG meeting for GreenMetadata ad hoc group, 2014. 8

[RNQ+15] E. Raffin, E. Nogues,, M. Pelcat and D. Menard HEVC Decoding with Tunable Image Quality - Power saving and complexity reduction, Contribution to 111th MPEG meeting for GreenMetadata ad hoc group, 2015. 8

[RLP+15] E. Raffin, M. Lacour, E. Nogues, M. Pelcat and D. Menard, Detailed Evaluation of Tunable Image Quality HEVC Decoding, Contribution to 112th MPEG meeting for GreenMetadata ad hoc group, 2015. 8

## National Conferences

[NRR+15] E. Nogues, E. Raffin, M. Pelcat and D. Menard, Décodeur vidéo HEVC basse consommation, XXVieme Colloque GRETSI, Septembre 2015

[NMP+14] E. Nogues, Maxime Pelcat, Daniel Ménard Poster: HEVC Decoding with Tunable Image Quality for Green Metadata applications Ecole thématique Conception FAible Consommation pour les systèmes embarqués et temps réels, May 2014

## Invited Presentations

[NNN+15] E. Nogues, DVFS adapté à la vidéo, Journée thématique du GDR ISIS et SoC SIP sur la consommation d'énergie dans les applications de traitement vidéo, October 2015

[NNP+14] E. Nogues, Démarche industrielle pour la conversion en virgule fixe, Journée thématique du GDR ISIS sur l'arithmétique pour le traitement de signal et de l'image, June 2014

# Bibliography


[ABDR13] Guillaume Aupy, Anne Benoit, Fanny Dufossé, and Yves Robert. Reclaiming the energy of a schedule: models and algorithms. *Concurrency and Computation: Practice and Experience*, 25(11):1505–1523, 2013. 50

[And98] Ray Andraka. A survey of cordic algorithms for fpga based computers. In *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, FPGA '98, pages 191–200, New York, NY, USA, 1998. ACM. 123

[ARM12] ARM. The ARM NEON general purpose SIMD. *www.arm.com/products/processors/technologies/neon.php*, 2012. 24, 40

[AVdS07] Emrah Akyol and Mihaela Van der Schaar. Complexity model based proactive dynamic voltage scaling for video decoding systems. *Multimedia, IEEE Transactions on*, 9(7):1475–1492, 2007. 99

[AVdS08] Emrah Akyol and Mihaela Van der Schaar. Compression-aware energy optimization for video decoding systems with passive power. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(9):1300–1306, 2008. 98

[BAMG+13] Benjamin Bross, Mauricio Alvarez-Mesa, Valeri George, Chi Ching Chi, Tobias Mayer, Ben Juurlink, and Thomas Schierl. Hevc real-time decoding. In *SPIE Optical Engineering+ Applications*, pages 88561R–88561R. International Society for Optics and Photonics, 2013. 5, 39, 40

[BBA11] Muhammad Khurram Bhatti, Cécile Belleudy, and Michel Auguin. Hybrid power management in real time embedded systems: an interplay of dvfs and dpm techniques. *Real-Time Systems*, 47(2):143–162, 2011. 50, 51

[bbc] BBC HEVC bistreams: . In *ftp://ftp.kw.bbc.co.uk/hevc/hm-15.0-anchors/*. 19, 85, 161

[BBS+15a] Yahia Benmoussa, Jalil Boukhobza, Eric Senn, Yassine Hadjadj-Aoul, and Djamel Benazzouz. A methodology for performance energy consumption characterization and modeling of video decoding on heterogeneous soc and its applications. *Journal of Systems Architecture*, 61(1):49 – 70, 2015. 65

[BBS+15b] Yahia Benmoussa, Jalil Boukhobza, Eric Senn, Yassine Hadjadj-Aoul, and Djamel Benazzouz. A methodology for performance/energy consumption characterization and modeling of video decoding on heterogeneous soc and its applications. *Journal of Systems Architecture*, 61(1):49–70, 2015. 97, 171

[BBSB13] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz. Energy consumption modeling of h.264/avc video decoding for gpp and dsp. In *Digital System Design (DSD), 2013 Euromicro Conference on*, pages 890–897, Sept 2013. 39

[BBSF12] Frank Bossen, Benjamin Bross, Karsten Suhring, and David Flynn. Hevc complexity and implementation analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1685–1696, 2012. 39, 40, 127, 136, 163

[BBYC09] Sung-Yong Bang, Kwanhu Bang, Sungroh Yoon, and Eui-Young Chung. Run-time adaptive workload estimation for dynamic voltage scaling. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(9):1334–1347, 2009. 98

[BD12] Luca Benini and Giovanni DeMicheli. *Dynamic power management: design techniques and CAD tools*. Springer Science & Business Media, 2012. 96, 121

[Ben15] Yahia Benmoussa. *Performance and Energy Consumption Characterization and Modeling of Video Decoding on Multi-core Heterogenous SoC and their Applications*. PhD thesis, Universite de Bretagne Sud, 2015. 39

[BJR11] George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons, 2011. 101

[BKVH07] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 8(1):67–127, 2007. 74

[Bos12] Frank Bossen. *Common Conditions and Software Reference Configurations*. Document JCTVC-H1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, San Jose, CA, Feb. 2012. 41, 145

[Bro05] Len Brown. Acpi in linux. In *Proceedings of the Linux Symposium, 51*, 2005. 30

[Bro13] Dominik Brodowski. Cpu frequency and voltage scaling code in the linux(tm) kernel. https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2013. 161

[BSB+14] Yahia Benmoussa, Eric Senn, Jalil Boukhobza, Mickael Lanoe, and Djamel Benazzouz. Open-people : A collaborative platform for remote & accurate measurement and evaluation of embedded systems power consumption. In *22nd IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2014. 64

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 62

[Cal96] Jean Paul Calvez. A codesign case study with the mcse methodology. *Design Automation for Embedded Systems*, 1(3):183–212, 1996. 49

[CAMJ+12] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schier. Parallel Scalability and Efficiency of HEVC Parallelization Approaches. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1827–1838, December 2012. 25

[CAMJ14] Chi Ching Chi, Mauricio Alvarez-Mesa, and Ben Juurlink. Low-power high-efficiency video decoding using general-purpose processors. *ACM Transactions on Architecture and Code Optimization*, 11(4):56, 2014. 5, 19, 33, 36, 40, 45, 96, 99, 101, 118, 160

[CAML+13a] Chi Ching Chi, Mauricio Alvarez-Mesa, Jan Lucas, Ben Juurlink, and Thomas Schierl. Parallel hevc decoding on multi-and many-core architectures. *Journal of Signal Processing Systems*, 71(3):247–260, 2013. 5, 26

[CAML+13b] ChiChing Chi, Mauricio Alvarez-Mesa, Jan Lucas, Ben Juurlink, and Thomas Schierl. Parallel hevc decoding on multi- and many-core architectures. *Journal of Signal Processing Systems*, 71(3):247–260, 2013. 96

[CCCY06] De-Shiuan Chiou, Shih-Hsin Chen, Shih-Chieh Chang, and Chingwei Yeh. Timing driven power gating. In *Proceedings of the 43rd annual design automation conference*, pages 121–124. ACM, 2006. 27

[CCKL11] Jinsung Cho, Ilyong Cho, Dae-Young Kim, and Ben Lee. A combined approach for qos-guaranteed and low-power video decoding. *Consumer Electronics, IEEE Transactions on*, 57(2):651–657, 2011. 154

[CCRR13] V.K. Chippa, S.T. Chakradhar, K. Roy, and A. Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *50th ACM/IEEE Design Automation Conference (DAC)*, pages 1–9, May 2013. 120

[CCS04] Naehyuck Chang, Inseok Choi, and Hojun Shim. Dls: dynamic backlight luminance scaling of liquid crystal display. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(8):837–846, 2004. 14

[CH89] JE Cooling and TS Hughes. The emergence of rapid prototyping as a real-time software development tool. In *Software Engineering for Real Time Systems, 1989., Second International Conference on*, pages 60–64. IET, 1989. 53

[CHK14] Gang Chen, Kai Huang, and Alois Knoll. Energy optimization for realtime multiprocessor soc with optimal dvfs and dpm combination. *ACM Trans. Embed. Comput. Syst.*, 13, 2014. 50, 51

[CHP04] Wei-Chung Cheng, Yu Hou, and Massoud Pedram. Power minimization in a backlit tft-lcd display by concurrent brightness and contrast scaling. In *Proceedings of the conference on Design, automation and test in Europe-Volume 1*, page 10252. IEEE Computer Society, 2004. 43

[CHP11] G. Clare, F. Henry, and S. Pateux. Wavefront Parellel Processing for HEVC Encoding and Decoding. In *document JCTVC-F274*. Torino, Italy, Jully 2011. 26

[Cis15] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper. February 2015. 3, 4, 169

[CJL10] S. Chevillard, M. Joldeş, and C. Lauter. Sollya: An environment for the development of numerical codes. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 28–31, Heidelberg, Germany, September 2010. Springer. 123

[CM10] Sangyeun Cho and Rami G Melhem. On the interplay of parallelization, program performance, and energy consumption. *Parallel and Distributed Systems, IEEE Transactions on*, 21(3):342–353, 2010. 51

[CMR+10] V.K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S.T. Chakradhar. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 555–560, June 2010. 121

[Cor07] Intel Corporation. Intel C++ Intrinsic Reference. 2007. 25

[CPG+13] M Chavarrias, F Pescador, MJ Garrido, M Raulet, et al. A dsp-based hevc decoder implementation using an actor language dataflow model. *Consumer Electronics, IEEE Transactions on*, 59(4):839–847, 2013. 5, 127, 136, 163

[CRRC11] V. Chippa, A. Raghunathan, K. Roy, and S. Chakradhar. Dynamic effort scaling: Managing the quality-efficiency tradeoff. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 603–608, June 2011. 120

[dDT01] F. de Dinechin and A. Tisserand. Some improvements on multipartite table methods. In *Proceedings. 15th IEEE Symposium on Computer Arithmetic, 2001*, pages 128–135, 2001. 123

[Des14] Karol Desnos. *Memory Study and Dataflow Representations for Rapid Prototyping of Signal Processing Applications on MPSoCs*. PhD thesis, INSA de Rennes, 2014. 53, 54, 76, 170

[DLJ06] Pepijn De Langen and Ben Juurlink. Leakage-aware multiprocessor scheduling for low power. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8–pp. IEEE, 2006. 50, 51, 59

[DSP66] Norman Richard Draper, Harry Smith, and Elizabeth Pownell. *Applied regression analysis*, volume 3. Wiley New York, 1966. 68

[DSY+14] Yizhou Duan, Jun Sun, Leju Yan, Keji Chen, and Zongming Guo. Novel efficient hevc decoding solution on general-purpose processors. 2014. 40

[Duc15] Xavier Ducloux. Green adaptive streaming. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 65. ACM, 2015. 44

[DVM12] Kai Du, Peter Varman, and Kartik Mohanram. High performance reliable variable latency carry select addition. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 1257–1262. IEEE, 2012. 120

[ea13] A. Tewari et al. Samsung's response to the call for proposals on green mpeg. *ISO/IEC/JTC1/SC29/WG11*, m30484(1), 2013. 43

[EL87] D. Messerschmitt E. Lee. Static scheduling of synchronous data-flow programs for digital signal processing. *IEEE Transactions on Computers*, pages 24–35, 1987. 50

[Ele] Samsung Electronics. Embedding content information in video streams for energy-efficient video processing on mobile devices. 42, 169

[exy13] Samsung GALAXY S4 Teardown. http://www.techinsights.com/inside-samsung-galaxy-s4/, 2013. 5, 41, 137

[FAA+12] Chih-Ming Fu, E. Alshina, A. Alshin, Yu-Wen Huang, Ching-Yeh Chen, Chia-Yang Tsai, Chih-Wei Hsu, Shaw-Min Lei, Jeong-Hoon Park, and Woo-Jin Han. Sample adaptive offset in the hevc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1755–1764, Dec 2012. 129

[FD15] Felix Fernandes and Xavier Ducloux. Report of the ahg on green mpeg. *ISO/IEC/JTC1/SC29/WG11*, MPEG 2015/m36758, 2015. 141

[FDM+15] Felix C Fernandes, Xavier Ducloux, Zhan Ma, Esmaeil Faramarzi, Patrick Gendron, and Jiangtao Wen. The green metadata standard for energy-efficient video consumption. *MultiMedia, IEEE*, 22(1):80–87, 2015. 8, 41, 43, 119, 169

[FFm] FFmpeg: Open source and cross-platform multimedia library. In *http://www.ffmpeg.org*. 99

[Fos94] Eric R Fossum. Ultra-low-power imaging systems using cmos image sensor technology. In *SPIE's 1994 International Symposium on Optics, Imaging, and Instrumentation*, pages 107–111. International Society for Optics and Photonics, 1994. 12

[Fra06] David J. Frank. The limits of cmos scaling from a power-constrained technology optimization perspective, Oct 2006. 21

[FUI] FUI. French Project GreenVideo, Available at http://greenvideo.insa-rennes.fr/. 8

[GBY08] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008. 62, 67, 83, 157

[GC97] Vadim Gutnik and Anantha P Chandrakasan. Embedded power supply for low-power dsp. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 5(4):425–435, 1997. 97

[GMMS97] Philip E. Gill, Walter Murray, Michael, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 1997. 56

[Gre13] Context, objectives, use cases and requirements for green mpeg. *SO/IEC JTC1/SC29/WG11/N13468*, April 2013. 6, 98, 153

[GS90] Stanton A Glantz and Bryan K Slinker. *Primer of applied regression and analysis of variance*. McGraw-Hill, 1990. 68

[GS03] Thierry Grandpierre and Yves Sorel. From algorithm and architecture specifications to automatic generation of distributed real-time executives: a seamless flow of graphs transformations. In *Formal Methods and Models for Co-Design, 2003. MEMOCODE'03. Proceedings. First ACM and IEEE International Conference on*, pages 123–132. IEEE, 2003. 53

[HALL13] Simon Holmbacka, Dag Agren, Sébastien Lafond, and Johan Lilius. Qos manager for energy efficient many-core operating systems. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 318–322. IEEE, 2013. 55

[HHL+13] Fredric Hällis, Simon Holmbacka, Wictor Lund, Robert Slotte, Sébastien Lafond, and Johan Lilius. Thermal influence on the energy efficiency of workload consolidation in many-core architectures. In *Digital Communications - Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*, pages 1–6, 2013. 27

[HK82] Frank Hansen and Gert Kjaerg. Complexity model based proactive dynamic voltage scaling for video decoding systems. *Mathematische Annalen*, 258(3):229–1241, 1982. 97

[HKG+13] Yuwen He, Markus Kunstner, Srinivas Gudumasu, Eun-Seok Ryu, Yan Ye, and Xiaoyu Xiu. Power aware hevc streaming for mobile. In *Visual Communications and Image Processing (VCIP), 2013*, pages 1–5. IEEE, 2013. 155

[HLC+05] Zhihai He, Yongfang Liang, Lulin Chen, Ishfaq Ahmad, and Dapeng Wu. Power-rate-distortion analysis for wireless video communication under energy constraints. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(5):645–658, 2005. 43

[HNP+14] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sébastien Lafond, and Johan Lilius. Energy efficiency and performance management of parallel dataflow applications. In *The 2014 Conference on Design & Architectures for Signal & Image Processing*, 2014. 7, 54, 55, 64, 71, 170

[HNP+15] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sébastien Lafond, Daniel Menard, and Johan Lilius. Energy-awareness and performance management with parallel dataflow applications. *Journal of Signal Processing Systems*, pages 1–16, 2015. 54

[HOB+12] Philipp Helle, Simon Oudin, Benjamin Bross, Detlev Marpe, M Oguz Bici, Kemal Ugur, Joel Jung, Gordon Clare, and Thomas Wiegand. Block merging

for quadtree-based partitioning in hevc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1720–1731, 2012. 18

[HPD+14] Julien Heulot, Maxime Pelcat, Karol Desnos, Jean-Francois Nezan, and Slaheddine Aridhi. Spider: A synchronous parameterized and interfaced dataflow-based rtos for multicore dsps. In *Education and Research Conference (EDERC), 2014 6th European Embedded Design in*, pages 167–171. IEEE, 2014. 53

[HRD14a] W. Hamidouche, M. Raulet, and O. Deforges. Parallel shvc decoder: Implementation and analysis. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6, July 2014. 26, 100, 169, 171

[HRD14b] W Hamidouche, M Raulet, and O Deforges. Parallel shvc decoder: Implementation and analysis. *IEEE conference on ICME*, 2014. 127, 154

[HRD14c] Wassim Hamidouche, Michael Raulet, and Olivier Déforges. Parallel shvc decoder: Implementation and analysis. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014. 38, 39, 40

[HRDSE12] Philippe Hanhart, Martin Rerabek, Francesca De Simone, and Touradj Ebrahimi. Subjective quality evaluation of the upcoming hevc video compression standard. In *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics, 2012. 131

[HZ10] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2366–2369. IEEE, 2010. 131

[Ins] Texas Instruments. INA231 Datasheet, Available at http://www.ti.com/product/INA231/technicaldocuments. 33

[IRF09] Mostafa EA Ibrahim, Markus Rupp, and Hossam AH Fahmy. Code transformations and simd impact on embedded software energy/power consumption. In *Computer Engineering & Systems, 2009. ICCES 2009. International Conference on*, pages 27–32. IEEE, 2009. 24

[IWC+01] Masaya Iwamoto, Aracely Williams, Pin-Fan Chen, Andre G Metzger, Lawrence E Larson, and Peter M Asbeck. An extended doherty amplifier with high efficiency over a wide power range. *Microwave Theory and Techniques, IEEE Transactions on*, 49(12):2472–2479, 2001. 14

[IY98] Tohru Ishihara and Hiroto Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pages 197–202. IEEE, 1998. 50

[JAS08] S. Irani J. Augustine and C. Swamy. Optimal power-down strategies, 2008. 29

[JMr] H.264/MPEG-4 AVC Reference software, joint model 18.6. http: iphome.hhi.de/suehring/tml/. 139

[JMSN05] Hailin Jiang, Malgorzata Marek-Sadowska, and Sani R Nassif. Benefits and costs of power-gating technique. In *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 559–566. IEEE, 2005. 23

[JPG04] Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st annual Design Automation Conference*, pages 275–280. ACM, 2004. 51, 58, 59, 64, 97

[JS15] C. Enz K. Palem J. Schlachter, V. Camus. Automatic generation of inexact digital circuits by gate-level pruning. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–6, May 2015. 120

[JW08] Tao Jiang and Yiyan Wu. An overview: peak-to-average power ratio reduction techniques for ofdm signals. *IEEE transactions on broadcasting*, 54(2):257, 2008. 13

[KAA+13] U Kemal, Alexander Alshin, Elena Alshina, Frank Bossen, W Han, J Park, and Jani Lainema. Motion compensated prediction and interpolation filter design in h. 265/hevc. 2013. 132, 133, 164

[KAB+03] Nam Sung Kim, Todd Austin, David Baauw, Trevor Mudge, Krisztián Flautner, Jie S Hu, Mary Jane Irwin, Mahmut Kandemir, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *computer*, 36(12):68–75, 2003. 20

[KAN+11] NEA Khalid, SA Ahmad, NM Noor, AFA Fadzil, and MN Taib. Parallel approach of sobel edge detector on multicore platform. *International Journal of Computers and Communications Issue*, 4:236–244, 2011. 57

[KK12] Andrew B Kahng and Seokhyeong Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th Annual Design Automation Conference*, pages 820–825. ACM, 2012. 120

[KP11] P.K. Krause and I. Polian. Adaptive voltage over-scaling for resilient applications. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011. 120

[KSY+02] Woonseok Kim, Dongkun Shin, Han-Saem Yun, Jihong Kim, and Sang-Lyul Min. Performance comparison of dynamic voltage scaling algorithms for hard real-time systems. In *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, pages 219–228, 2002. 50, 97, 121

[LBH+12] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min, and Kemal Ugur. Intra coding of the hevc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1792–1801, 2012. 18

[LCW+15] Tsu-Ming Liu, Yung-Chang Chang, Chih-Ming Wang, Hue-Min Lin, Chia-Yun Cheng, Chun-Chia Chen, Min-Hao Chiu, Sheng-Jen Wang, Ping Chao, Meng-Jye Hu, et al. Energy and area efficient hardware implementation of

4k main-10 hevc decoder in ultra-hd blu-ray player and tv systems. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015. 36, 38, 45, 96

[LEN+11] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011. 120

[LFCM07] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. Gpac: open source multimedia framework. In *Proceedings of the 15th international conference on Multimedia*, pages 1009–1012. ACM, 2007. 118

[LLSS03] Zhijian Lu, John Lach, Mircea Stan, and Kevin Skadron. Reducing multimedia decode power using feedback control. In *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 489–496. IEEE, 2003. 97

[LNC96] J.T. Ludwig, S.H. Nawab, and A.P. Chandrakasan. Low-power digital filtering using approximate processing. *IEEE Journal of Solid-State Circuits*, 31(3):395–400, Mar 1996. 121

[Lue97] David G Luenberger. *Optimization by vector space methods*. John Wiley & Sons, 1997. 74

[LWX+12] Hao Lv, Ronggang Wang, Xiaodong Xie, Huizhu Jia, and Wen Gao. A comparison of fractional-pel interpolation filters in hevc and h. 264/avc. In *VCIP*, pages 1–6, 2012. 132

[Mat12] MathWorks, Inc., Natick, Massachusetts, United States. MATLAB and Statistics Toolbox Release 2012b, 2012. 49

[MBM+98] Thomas B Mader, Eric W Bryerton, Milica Marković, Michael Forman, and Zoya Popović. Switched-mode high-efficiency microwave power amplifiers in a free-space power-combiner array. *Microwave Theory and Techniques, IEEE Transactions on*, 46(10):1391–1398, 1998. 14

[MCRR11] D. Mohapatra, V.K. Chippa, A. Raghunathan, and K. Roy. Design of voltage-scalable meta-functions for approximate computing. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011. 120

[MGG+11] Saeed Maleki, Yaoqing Gao, Mara J Garzaran, Tommy Wong, David Padua, et al. An evaluation of vectorizing compilers. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*, pages 372–382. IEEE, 2011. 25

[MHW11] Zhan Ma, Hao Hu, and Yao Wang. On complexity modeling of h. 264/avc video decoding and its application for energy efficient decoding. *Multimedia, IEEE Transactions on*, 13(6):1240–1255, 2011. 98, 99

[MJR+13] Gaurav Mitra, Benjamin Johnston, Alistair P Rendell, Eric McCreath, and Jun Zhou. Use of simd vector operations to accelerate application code

performance on low-powered arm and intel platforms. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1107–1116. IEEE, 2013. 24

[MKA+13] N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. van der Schaar. Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems. *Multimedia, IEEE Transactions on*, 15(2):268–278, 2013. 154

[MPE] MPEG. Information technology - mpeg systems technolo- gies - part 11: Energy-efficient media consumption (green metadata). 41

[MRZ+03] Ramesh Mishra, Namrata Rastogi, Dakai Zhu, Daniel Mossé, and Rami Melhem. Energy aware scheduling for distributed real-time systems. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 9–pp. IEEE, 2003. 50

[Mud01] Trevor Mudge. Computer power: A first-class architectural design constraint. *IEEE Computer Magazine*, 34:58–52, 2001. 20, 153

[NBF+12] Andrey Norkin, Gisle Bjontegaard, Arild Fuldseth, Matthias Narroschke, Masaru Ikeda, Kenneth Andersson, Minhua Zhou, and Geert Van der Auwera. Hevc deblocking filter. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1746–1754, 2012. 18

[NBP+15] Erwan Nogues, Romain Berrada, Maxime Pelcat, Daniel Menard, and Erwan Raffin. A dvfs based hevc decoder for energy-efficient software implementation on embedded processors. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015. 8, 19, 36, 37, 84, 106, 169

[NHP+14] Erwan Nogues, Simon Holmbacka, Maxime Pelcat, Daniel Menard, and Johan Lilius. Power-aware hevc decoding with tunable image quality. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pages 1–6. IEEE, 2014. 8, 33, 38, 129, 155

[NM14] Erwan Nogues and Daniel Menard. Efficient fixed-point refinement of dsp dataflow systems. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pages 1–6. IEEE, 2014. 93

[NMM+11] Andrew Nelson, Orlando Moreira, Anca Molnos, Sander Stuijk, Ba Thang Nguyen, and Kees Goossens. Power minimisation for real-time dataflow applications. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 117–124. IEEE, 2011. 50, 51, 59

[NRPM15a] Erwan Nogues, Erwan Raffin, Maxime Pelcat, and Daniel Ménard. Décodeur video hevc basse consommation. In *XXVème colloque GRETSI*, 2015. 8

[NRPM15b] Erwan Nogues, Erwan Raffin, Maxime Pelcat, and Daniel Menard. Hevc decoding with tunable image quality-subjective evaluation. 2015. 139, 146

[NRPM15c] Erwan Nogues, Erwan Raffin, Maxime Pelcat, and Daniel Menard. A modified hevc decoder for low power decoding. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 60. ACM, 2015. 8, 129

[NT] Erwan Nogues and Teamcast. French Project GreenVideo - Livrable D1.2 - Elements de la consommation dans chaîne de traiement vidéo, Available at http://greenvideo.insa-rennes.fr/. 13

[OBL+04] Jörn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer, and Thomas Wedi. Video coding with h. 264/avc: tools, performance, and complexity. *Circuits and Systems magazine, IEEE*, 4(1):7–28, 2004. 16

[Ogu90] P. Macken M. Degrauwe M. V. Paemel H. Oguey. A voltage reduction technique for digital systems, feb 1990. 29

[Ope] The Open HEVC - open source project. https://github.com/OpenHEVC/openHEVC. 6, 41, 99, 107, 128, 134, 136, 161

[OSS+12a] J Ohm, Gary J Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. Comparison of the coding efficiency of video coding standards—including high efficiency video coding (hevc). *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1669–1684, 2012. 130, 139

[OSS+12b] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand. Comparaison of the Coding Efficiency of Video Coding standards including High Efficiency Video coding (HEVC). *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1858–1870, December 2012. 18

[p9192] P.910: Subjective video quality assessment methods for multimedia applications, ITU-R (1992). 131

[PBR09] Jonathan Piat, Shuvra S Bhattacharyya, and Mickaël Raulet. Interface-based hierarchy for synchronous data-flow graphs. In *Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on*, pages 145–150. IEEE, 2009. 52

[PDH+14] Maxime Pelcat, Karol Desnos, Julien Heulot, Clément Guy, Jean-François Nezan, and Slaheddine Aridhi. Preesm: A dataflow-based rapid prototyping framework for simplifying multicore dsp programming. In *Education and Research Conference (EDERC), 2014 6th European Embedded Design in*, pages 36–40. IEEE, 2014. 54, 155

[Pig04] Christian Piguet. *Low-power electronics design*. CRC press, 2004. 58

[PLB07] Venkatesh Pallipadi, Shaohua Li, and Adam Belay. cpuilde - do nothing efficiently, ... In *Proceedings of the Linux Symposium*, 2007. 29, 31

[PMvKS95] Christian Piguet, J-M Masgonty, V von Kaenel, and T Schneider. Logic design for low-voltage/low-power cmos circuits. In *Proceedings of the 1995 international symposium on Low power design*, pages 117–122. ACM, 1995. 23, 50

[Poi] Mathieu Poirier. In kernel switcher: A solution to support arm's new big.little technology. *Embedded Linux Conference*. 104

[PPW+09] Maxime Pelcat, Jonathan Piat, Matthieu Wipliez, Slaheddine Aridhi, and Jean-François Nezan. An open framework for rapid prototyping of signal processing applications. *EURASIP journal on embedded systems*, 2009:11, 2009. 53

[PRMS10] K. Parashar, R. Rocher, D. Menard, and O. Sentieys. A Hierarchical Methodology for Word-Length Optimization of Signal Processing Systems. In *Proc. International Conference on VLSI Design*, Bangalore, January 2010. 124

[PS06] Venkatesh Pallipadi and Alexey Starikovskiy. The ondemand governor. In *Proceedings of the Linux Symposium*, 2006. 138

[RCN02] Jan M Rabaey, Anantha P Chandrakasan, and Borivoje Nikolic. *Digital integrated circuits*, volume 2. Prentice hall Englewood Cliffs, 2002. 23, 58

[RHD+10] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010. 16

[Ric97] Mark A Richards. *Rapid Prototyping of Application Specific Signal Processors*. Springer Science & Business Media, 1997. 52

[RJS+10] R Ren, Eduardo Juarez, Cesar Sanz, Michael Raulet, and Fernando Pescador. An analysis of power consumption in a smartphone. *Proceedings of the USENIX Annual Technical Conference*, pages 21–28, 2010. 5

[RJS+13] Rong Ren, Eduardo Juarez, Cesar Sanz, Michael Raulet, and Fernando Pescador. System-level pmc-driven energy estimation models in rvc-cal video codec specifications. In *Design and Architectures for Signal and Image Processing (DASIP), 2013 Conference on*, pages 55–62. IEEE, 2013. 64

[RJS+14] R Ren, Eduardo Juarez, Cesar Sanz, Michael Raulet, and Fernando Pescador. Energy-aware decoder management: a case study on rvc-cal specification based on just-in-time adaptive decoder engine. *Consumer Electronics, IEEE Transactions on*, 60(3):499–507, 2014. 65

[RLdS+09] Barry Rountree, David K. Lownenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, and Tyler Bletsch. Adagio: Making dvs practical for complex hpc applications. In *Proceedings of the 23rd International Conference on Supercomputing*, ICS '09, pages 460–469, New York, NY, USA, 2009. ACM. 27

[RMMM03] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits. *Proceedings of the IEEE*, 91(2):305–327, Feb 2003. 20

[RNH+15] Erwan Raffin, Erwan Nogues, Wassim Hamidouche, Seppo Tomperi, Maxime Pelcat, and Daniel Menard. Low power hevc software decoder for mobile devices. *Journal of Real-Time Image Processing*, pages 1–13, 2015. 5, 8, 25, 26, 33, 36, 39, 40, 96, 101, 104, 127, 169

[RNPM15] Erwan Raffin, Erwan Nogues, Maxime Pelcat, and Daniel Menard. Hevc decoding with tunable image quality-power saving and complexity reduction. 2015. 146

[Roc70] R Tyrrell Rockafellar. Convex analysis (princeton mathematical series). *Princeton University Press*, 46:49, 1970. 66

[RR12] T. Rauber and G. Runger. Energy-aware execution of fork-join-based task parallelism. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 231–240, 2012. 27

[RTM+10] Stefan Rusu, Simon Tam, Harry Muljono, Jason Stinson, David Ayers, Jonathan Chang, Raj Varada, Matt Ratta, Sailesh Kottapalli, and Sujal Vora. A 45 nm 8-core enterprise xeon processor. *Solid-State Circuits, IEEE Journal of*, 45(1):7–14, 2010. 23

[RWB09] Krishna K Rangan, Gu-Yeon Wei, and David Brooks. Thread motion: fine-grained power management for multi-core systems. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 302–313. ACM, 2009. 104

[RWJ+13] Rong Ren, Jianguo Wei, Eduardo Juarez, Matias Garrido, Cesar Sanz, and Fernando Pescador. A pmc-driven methodology for energy estimation in rvc-cal video codec specifications. *Signal Processing: Image Communication*, 28(10):1303–1314, 2013. 65

[Sam12a] Embedding content information in video streams for energy-efficient video processing on mobile devices. *ISO/IEC JTC1/SC29/WG11 MPEG2012/*, April 2012. 14

[Sam12b] Samsung. Evaluation on exynos.bl processor. In *Korea Linux Forum*. Linux Foundation, 2012. 104

[SB12] Vivienne Sze and Madhukar Budagavi. High throughput cabac entropy coding in hevc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1778–1791, 2012. 18

[SBS14] Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. *High Efficiency Video Coding (HEVC)*. Springer, 2014. 100

[SIA] International Technology Roadmap for Semiconductors. http://public.itrs.net. 21

[SKCP11] Donghwa Shin, Younghyun Kim, Naehyuck Chang, and Massoud Pedram. Dynamic voltage scaling of oled displays. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 53–58. IEEE, 2011. 14, 42

[SKH95] Behrooz A Shirazi, Krishna M Kavi, and Ali R Hurson. *Scheduling and load balancing in parallel and distributed systems*. IEEE Computer Society Press, 1995. 78

[SKL01] Dongkun Shin, Jihong Kim, and Seongsoo Lee. Intra-task voltage scheduling for low-energy, hard real-time applications. *IEEE Design & Test of Computers*, (2):20–30, 2001. 50

[SM98] David K Su and William J McFarland. An ic for linearizing rf power amplifiers using envelope elimination and restoration. *Solid-State Circuits, IEEE Journal of*, 33(12):2252–2258, 1998. 13

[SMW07] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9):1103–1120, 2007. 38

[SNPF04] Christian Schuster, Jean-Luc Nagel, Christian Piguet, and Pierre-André Farine. Leakage reduction at the architectural level and its application to 16 bit multiplier architectures. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pages 169–178. Springer, 2004. 21, 22, 23, 169, 173

[Sod11] Iraj Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, (4):62–67, 2011. 6, 43

[SOHW12a] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. Overview of the high efficiency video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1648–1667, December 2012. 18, 99

[SOHW12b] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012. 37, 126, 129, 133, 163

[SS83] AAM Saleh and J Salz. Adaptive linearization of power amplifiers in digital radio systems. *Bell System Technical Journal*, 62(4):1019–1033, 1983. 13

[SS95] Ralf Schafer and Thomas Sikora. Digital video coding standards and their role in video communications. *Proceedings of the IEEE*, 83(6):907–924, 1995. 15

[SSM14] Heiko Schwarz, Thomas Schierl, and Detlev Marpe. Block structures and parallelism features in hevc. In *High Efficiency Video Coding (HEVC)*, pages 49–90. Springer, 2014. 18

[TL08] Chien-Cheng Tseng and Su-Ling Lee. Design of fractional delay fir filter using discrete cosine transform. In *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pages 858–861. IEEE, 2008. 132, 164

[TMQW06] Ying Tan, Parth Malani, Qinru Qiu, and Qing Wu. Workload prediction and dynamic voltage scaling for mpeg decoding. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pages 911–916. IEEE Press, 2006. 98

[Tom06] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006. 85

[VC15] C. Enz V. Camus, J. Schlachter. Energy-efficient inexact speculative adder with high performance and accuracy control. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–6, May 2015. 120

[VEWT96] P Van Eetvelt, G Wade, and M Tomlinson. Peak to average power reduction for ofdm schemes by selective scrambling. *Electronics letters*, 32(21):1963–1964, 1996. 13

[VQE08] Video Quality Experts Group (VQEG), Final report of video quality experts group multimedia phase I validation test, 2008. 130

[Wat] Amos Waterland. The STRESS benchmark, Available at http://people.seas.harvard.edu/ apw/stress/. 64

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004. 130

[Wei84] Reinhold P Weicker. Dhrystone: a synthetic systems programming benchmark. *Communications of the ACM*, 27(10):1013–1030, 1984. 104

[WLL+13] Jiangtao Wen, Bohan Li, Shunyao Li, Yao Lu, and Pin Tao. Cross segment decoding of hevc for network video applications. In *Packet Video Workshop (PV), 2013 20th International*, pages 1–8. IEEE, 2013. 44

[WPW00] Qing Wu, Massoud Pedram, and Xunwei Wu. Clock-gating and its application to low power design of sequential circuits. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 47(3):415–420, 2000. 23

[WSBL03] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuit and Systems for Video Technology*, 13(7):560–576, July 2003. 18

[WSJ+03] Thomas Wiegand, Heiko Schwarz, Anthony Joch, Faouzi Kossentini, and Gary J Sullivan. Rate-constrained coder control and comparison of video coding standards. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):688–703, 2003. 37

[WSV+05] Clemens C Wüst, Liesbeth Steffens, Wim FJ Verhaegh, Reinder J Bril, and Christian Hentschel. Qos control strategies for high-quality video processing. *Real-Time Systems*, 30(1-2):7–29, 2005. 32

[WYK+05] Feipeng Wang, Annie Hueiching Yang, Donald F Kimball, Lawrence E Larson, and Peter M Asbeck. Design of wide-bandwidth envelope-tracking power amplifiers for ofdm applications. *Microwave Theory and Techniques, IEEE Transactions on*, 53(4):1244–1255, 2005. 14

[YDS95] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 374. IEEE Computer Society, 1995. 97

[You12] Embedding Content Information in Video Streams for Energy-efficient Video Processing on Mobile Devices. *ISO/IEC JTC1/SC29/WG11 MPEG2012/m24982*, Geneva, April 2012. 101

[ZbYAd11] Du Zhi-bo, Chen Yun, and Chen Ai-dong. The impact of the clock frequency on the power analysis attacks. In *Internet Technology and Applications (iTAP), 2011 International Conference on*, pages 1–4, 2011. 27

[ZLC+03] Ana Luíza Zuquim, Antonio A. F. Loureiro, Claudionor J. N. Coelho Jr., Marcos Augusto M. Vieira, Luiz Filipe M. Vieira, Alex Borges Vieira, Antônio O. Fernandes, Diógenes C. da Silva Jr., José M. da Mata, José Augusto Nacif, and Hervaldo Sampaio. Efficient power management in real-time embedded systems, September 2003. 28, 169

[ZZH+11] Dajiang Zhou, Jinjia Zhou, Xun He, Jiayi Zhu, Ji Kong, Peilin Liu, and Satoshi Goto. A 530 mpixels/s 4096x2160@ 60fps h. 264/avc high profile video decoder chip. *Solid-State Circuits, IEEE Journal of*, 46(4):777–788, 2011. 38

INSA de RENNES
Service des Formations

# AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

**Titre de la thèse:**
Energy Optimization of Signal Processing on MPSoCs and its Application to Video Decoding

**Nom Prénom de l'auteur : NOGUES ERWAN**

Membres du jury :
- Monsieur MENARD Daniel
- Monsieur PELCAT Maxime
- Monsieur GRANADO Bertrand
- Monsieur CASSEAU Emmanuel
- Madame BELLEUDY Cécile
- Monsieur SENN Eric
- Monsieur JUAREZ Eduardo
- Monsieur GINHAC Dominique

Président du jury : E. CASSEAU

Date de la soutenance : 02 Juin 2016

Reproduction de la these soutenue

Thèse pouvant être reproduite en l'état
~~Thèse pouvant être reproduite après corrections suggérées~~

Fait à Rennes, le 02 Juin 2016

Signature du président de jury

Le Directeur,

M'hamed DRISSI

## Résumé

Aujourd'hui, les appareils électroniques offrent de plus en plus de fonctionnalités (vidéo, audio, GPS, internet) et des connectivités variées (multi-systèmes de radio avec WiFi, Bluetooth, UMTS, HSPA, LTE-advanced ...). La demande en puissance de ces appareils est donc grandissante pour la partie numérique et notamment le processeur de calcul. Pour répondre à ce besoin sans cesse croissant de nouvelles fonctionnalités et donc de puissance de calcul, les architectures des processeurs ont beaucoup évolué : processeurs multi-coeurs, processeurs graphiques (GPU) et autres accélérateurs matériels dédiés. Cependant, alors que de nouvelles architectures matérielles peinent à répondre aux exigences de performance, l'évolution de la technologie des batteries est quant à elle encore plus lente. En conséquence, l'autonomie des systèmes embarqués est aujourd'hui sous pression.

Parmi les nouveaux services supportés par les terminaux mobiles, la vidéo prend une place prépondérante. En effet, des analyses récentes de tendance montrent qu'elle représentera 70 % du trafic internet mobile dès 2016. Accompagnant cette croissance, de nouvelles technologies émergent permettant de nouveaux services et applications. Parmi elles, HEVC (High Efficiency Video Coding) permet de doubler la compression de données tout en garantissant une qualité subjective équivalente à son prédécesseur, la norme H.264.

Dans un circuit numérique, la consommation provient de deux éléments: la puissance statique et la puissance dynamique. La plupart des architectures matérielles récentes mettent en œuvre des procédés permettant de contrôler la puissance du système. Le changement dynamique du couple tension/fréquence appelé Dynamic Voltage and Frequency Scaling (DVFS) agit principalement sur la puissance dynamique du circuit. Cette technique permet d'adapter la puissance du processeur (et donc sa consommation) à la charge réelle nécessaire pour une application. Pour contrôler la puissance statique, le Dynamic Power Management (DPM, ou modes de veille) consistant à arrêter les alimentations associées à des zones spécifiques de la puce.

Dans cette thèse, nous présentons d'abord une modélisation de l'énergie consommée par le circuit intégrant les modes DVFS et DPM. Cette modélisation est généralisée au circuit multi-coeurs et intégrée à un outil de prototypage rapide. Ainsi le point de fonctionnement optimal d'un circuit, la fréquence de fonctionnement et le nombre de cœurs actifs, est identifié. Dans un second temps, l'application HEVC est intégrée à une architecture multi-cœurs avec une adaptation dynamique de la fréquence de développement. Nous montrons que cette application peut être implémentée efficacement sur des processeurs généralistes (GPP) tout en minimisant la puissance consommée. Enfin, et pour aller plus loin dans les gains en énergie, nous proposons une modification du décodeur HEVC qui permet à un décodeur de baisser encore plus sa consommation en fonction du budget énergétique disponible localement.

## Abstract

Consumer electronics offer today more and more features (video, audio, GPS, Internet) and connectivity means (multi-radio systems with WiFi, Bluetooth, UMTS, HSPA, LTE-advanced ...). The power demand of these devices is growing for the digital part especially for the processing chip. To support this ever increasing computing demand, processor architectures have evolved with multicore processors, graphics processors (GPU) and other dedicated hardware accelerators. However, the evolution of battery technology is itself slower. Therefore, the autonomy of embedded systems is now under a great pressure.

Among the new functionalities supported by mobile devices, video services take a prominent place. Indeed, recent analyzes show that they will represent 70% of mobile Internet traffic by 2016. Accompanying this growth, new technologies are emerging for new services and applications. Among them HEVC (High Efficiency Video Coding) can double the data compression while maintaining a subjective quality equivalent to its predecessor, the H.264 standard.

In a digital circuit, the total power consumption is made of static power and dynamic power. Most of modern hardware architectures implement means to control the power consumption of the system. Dynamic Voltage and Frequency Scaling (DVFS) mainly reduces the dynamic power of the circuit. This technique aims to adapt the power of the processor (and therefore its consumption) to the actual load needed by the application. To control the static power, Dynamic Power Management (DPM or sleep modes) aims to stop the voltage supplies associated with specific areas of the chip.

In this thesis, we first present a model of the energy consumed by the circuit integrating DPM and DVFS modes. This model is generalized to multi-core integrated circuits and to a rapid prototyping tool. Thus, the optimal operating point of a circuit, i.e. the operating frequency and the number of active cores, is identified. Secondly, the HEVC application is integrated to a multicore architecture coupled with a sophisticated DVFS mechanism. We show that this application can be implemented efficiently on general purpose processors (GPP) while minimizing the power consumption. Finally, and to get further energy gain, we propose a modified HEVC decoder that is capable to tune its energy gains together with a decoding quality trade-off.



INSA RENNES

N° d'ordre : **16ISAR 09 / D16 - 09**


**Institut National des Sciences Appliquées de Rennes**
20, Avenue des Buttes de Coësmes - CS 14315 - F-35043 Rennes Cedex
Tél : 02 23 23 82 00 – Fax : 02 23 23 83 96